2019

Knowledge-Enabled Entity Extraction

Hussein S. Al-Olimat
Wright State University

KNOWLEDGE-ENABLED ENTITY EXTRACTION

A Dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

by

HUSSEIN S. AL-OLIMAT
B.S., German Jordanian University, 2012
M.S., The University of Toledo, 2014

2019
Wright State University

Wright State University
GRADUATE SCHOOL

November 19, 2019

I HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER MY SUPERVISION BY HUSSEIN S. AL-OLIMAT ENTITLED KNOWLEDGE-ENABLED ENTITY EXTRACTION BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

Krishnaprasad Thirunarayan, Ph.D.
Dissertation Director

Michael Raymer, Ph.D.
Director, Computer Science and Engineering Ph.D. Program

Barry Milligan, Ph.D.
Interim Dean of the Graduate School

Committee on
Final Examination

Krishnaprasad Thirunarayan, Ph.D.

Keke Chen, Ph.D.

Guozhu Dong, Ph.D.

Steven Gustafson, Ph.D.

Srinivasan Parthasarathy, Ph.D.

Valerie L. Shalin, Ph.D.

ABSTRACT

AL-OLIMAT, HUSSEIN S. PhD., Department of Computer Science and Engineering, Wright State University, 2019. *KNOWLEDGE-ENABLED ENTITY EXTRACTION.*

Information Extraction (IE) techniques are developed to extract entities, relationships, and other detailed information from unstructured text. The majority of the methods in the literature focus on designing supervised machine learning techniques, which are not very practical due to the high cost of obtaining annotations and the difficulty in creating high quality (in terms of reliability and coverage) gold standard. Therefore, semi-supervised and distantly-supervised techniques are getting more traction lately to overcome some of the challenges, such as bootstrapping the learning quickly.

This dissertation focuses on information extraction, and in particular entities, i.e., Named Entity Recognition (NER), from multiple domains, including social media and other grammatical texts such as news and medical documents. This work explores the ways for lowering the cost of building NER pipelines with the help of available knowledge without compromising the quality of extraction and simultaneously taking into consideration feasibility and other concerns such as user-experience. I present a type of distantly supervised (dictionary-based), supervised (with reduced cost using entity set expansion and active learning), and minimally-supervised NER approaches. In addition, I discuss the various aspects of the knowledge-enabled NER approaches and how and why they are a better fit for today's real-world NER pipelines in dealing with and partially overcoming the above-mentioned difficulties.

I present two dictionary-based NER approaches. The first technique extracts location mentions from text streams, which proved very effective for stream processing with competitive performance in comparison with ten other techniques. The second is a generic NER approach that scales to multiple domains and is minimally supervised with a human-in-the-loop for online feedback. The two techniques augment and filter the dictionaries to compensate for their incompleteness (due to lexical variation between dictionary records and mentions in the text) and for eliminating the noise and spurious content in them. The

third technique I present is a supervised approach but with a reduced cost in terms of the number of labeled samples and the complexity of annotating. The cost reduction was achieved with the help of a human-in-the-loop and smart instance samplers implemented using entity set expansion and active learning. The use of knowledge, the monitoring of NER models' accuracy, and the full exploitation of inputs from the human-in-the-loop was the key to overcoming the practical and technical challenges. I make the data and code for the approaches presented in this dissertation publicly available.

Contents

1 Introduction **1**
- 1.1 Overview . . . 1
  - 1.1.1 Knowledge Sources, Creation, and Reuse . . . 3
  - 1.1.2 NER from Text Streams using Dictionaries . . . 5
  - 1.1.3 User-centered and Cost-effective NER . . . 10
  - 1.1.4 From Candidate Entities To Named Entities . . . 15
- 1.2 Thesis Statement . . . 18
- 1.3 Research Questions . . . 18
- 1.4 Contributions . . . 19
- 1.5 Dissertation Structure . . . 21

2 Location Name Extraction **22**
- 2.1 Introduction . . . 23
- 2.2 Challenges of Location Name Extraction . . . 24
- 2.3 Approach . . . 27
  - 2.3.1 Statistical Inference via n-gram Models . . . 28
  - 2.3.2 Gazetteer Augmentation and Filtering . . . 30
  - 2.3.3 Tweet Preprocessing . . . 33
  - 2.3.4 Extracting Location Names using LNEx . . . 35
  - 2.3.5 Time and Space Complexity . . . 36
- 2.4 Evaluation . . . 37
  - 2.4.1 Data Collection . . . 37
  - 2.4.2 Data Filtering and Preparation . . . 38
  - 2.4.3 Benchmarking and Annotations . . . 39
  - 2.4.4 Evaluation Strategy . . . 41
  - 2.4.5 Spell Checking . . . 42
  - 2.4.6 Hashtag Breaking . . . 42
  - 2.4.7 Picking a Gazetteer . . . 43
  - 2.4.8 Comparing LNEx with Other Tools . . . 43
  - 2.4.9 Illustrative Examples . . . 48
- 2.5 Related Work . . . 51
- 2.6 Conclusions, Limitations, and Next Steps . . . 53

3 Sparse Entity Extraction **55**
3.1 Introduction . . . 56
3.2 Incremental and Interactive Learning . . . 57
3.2.1 Active Learning (AL) . . . 57
3.2.2 Annotation Modes . . . 59
3.2.3 Interactive Learning . . . 60
3.3 Entity Set Expansion for Base Model Learning . . . 61
3.3.1 Noun Phrase Extraction (NPEx) . . . 62
3.3.2 Featurization . . . 63
3.3.3 Feature-Graph Embedding . . . 64
3.3.4 Set Expansion using Graph Embedding . . . 66
3.3.5 Feature Ensemble Ranking . . . 67
3.4 Experiments and Results . . . 67
3.4.1 Data Preparation . . . 68
3.4.2 Noun Phrase Extraction Evaluation . . . 69
3.4.3 Entity Set Expansion (ESE) Evaluation . . . 69
3.4.4 Full System Evaluation . . . 72
3.5 Related Work . . . 78
3.6 Conclusions, Limitations, and Next Steps . . . 79

4 Minimally Supervised Entity Extraction **80**
4.1 Introduction . . . 80
4.2 Candidate Entity Extraction . . . 82
4.2.1 Language Model-based Extractor (LMEx) . . . 82
4.2.2 Regular Expression-based Extractor (RegEx) . . . 83
4.3 Minimal Supervision . . . 84
4.3.1 Candidate Entity Featurization . . . 85
4.3.2 Pattern-based Candidate Entity Ranking . . . 88
4.3.3 Partial Labels Exploitation . . . 89
4.4 Multiview-based Classification . . . 89
4.5 Experiments . . . 91
4.5.1 Experimental Settings . . . 91
4.5.2 Performance Evaluation . . . 98
4.6 Related Work . . . 105
4.7 Conclusions, Limitations, and Future Steps . . . 107

5 Conclusions and Future Directions **108**

Bibliography **112**

List of Figures

1.1 Two SME-in-the-loop labeling techniques: (a) disease mentions span labeling and (b) cancer concept binary labeling. 11

2.1 Extracting locations using the LNEx tool. 35
2.2 Example location name annotations using the BRAT tool in the context of the 2016 Louisiana flood. 40
2.3 2016 Louisiana floods map: the 21 Louisiana parishes that were declared federal disaster areas by FEMA (Source [130]). 40
2.4 Hashtag breaking accuracy. 42
2.5 Raw vs. augmented and filtered (AF) gazetteers combinations. Each of the seven combinations is a subset of {OSM Geonames DBpedia}. 45
2.6 Gazetteer combinations performance. Each of the seven combinations is a subset of {OSM Geonames DBpedia}. E.g., 100 stands for {OSM} while 011 stands for {Geonames DBpedia}. 45
2.7 Random sample evaluation showing the prevalence of the challenges and the frequency of the different form types discussed in Section 2.2. 46

3.1 The three stages of graph embedding for candidate entities ranking. 66
3.2 Coarse features ensemble method for ranking candidate entities. 67
3.3 End-to-End system pipeline. Arrows represent paths that can be followed to annotate the text. 71
3.4 Percentage of sentences annotated while using EAL to reach different F-Scores. 72
3.5 Learning curves of the approach on two entity classes from the CoNLL 2003 dataset with different querying strategies. Y-axis value for AR and EAL is the F-Score, and for Sigma confidence (σ) and Estimated Coverage is the value from Equations 3.3 and 3.5, respectively. 74
3.6 Learning curves of the approach on two entity classes from the GENIA dataset with different querying strategies. Y-axis value for AR and EAL is the F-Score, and for Sigma confidence (σ) and Estimated Coverage is the value from Equations 3.3 and 3.5, respectively. 75

3.7 Learning curves of the approach on two entity classes one from the GENIA dataset and the second from BioCreAtIvE II with different querying strategies. Y-axis value for AR and EAL is the F-Score, and for Sigma confidence (σ) and Estimated Coverage is the value from Equations 3.3 and 3.5, respectively. 76

4.1 The system architecture of our adaptive knowledge-enabled pipeline (KnowEx). 92

List of Tables

2.1 Example challenges and types of location names in the social media text. . 25
2.2 Gazetteer augmentation to generate alternative names. 31
2.3 Extraneous text in raw gazetteers. 32
2.4 Statistics of location mentions inside hashtags, where, **# UHs** is the number of unique hashtags, **UHs*** are the unique hashtags excluding the ones used for crawling. 33
2.5 Types considered as location classes per tool. 45
2.6 Location extraction tools vs. LNEx with a raw (RawGaz), and augmented and filtered gazetteer (AFGaz). LNEx improves its performance by using AFGaz and outperforms all tools with at least 33% F1-Score improvement. 47
2.7 Example tool outputs: the bracketed bold text is the identified location names, and braces highlight the types from Section 2.2. 49

3.1 Datasets statistics showing the entity classes, the number of sentences ($|S|$) containing entities, and the number of entities across all sentences. 68
3.2 Noun phrase extraction performance. 69
3.3 The frequencies of the seed entities (in all sentences) used in the evaluation of the ESE method. One high-frequent and one low-frequent seed entities were used to test the effectiveness of the ranking mechanism per dataset. . . 70
3.4 ESE performance ($p@k$) while using the feature ensemble method. The best performing combination is boldfaced. 70
3.5 Pipeline testing results of the EAL and EAA annotation modes showing the model confidence (σ), F-Scores, and percentage cut from the pool of sentences. 73

4.1 Datasets statistics: The number of sentences and entities in each corpus (C# 1 to 10). 93
4.2 The aggregated weights of Wikipedia categories for the domain-specific dictionaries. 96
4.3 Performance of the dictionary-based entity extractor (LMEx) while using raw and filtered dictionaries. Bold-faced is for the improved precision while the underlined show the ones without a difference. 97

4.4 Performance of the dictionary-based entity extractor (LMEx) with filtered dictionaries vs. AutoNER. Bold-faced values reflect the best performances. 97
4.5 Precision at k (P@k) and the number of instances covered in sentences (Instance Count) for the best performing ranking mechanism (lsv) on LMEx extractions. SMEs-in-the-loop label at most 50 instances. 99
4.6 The number of candidate entities extracted using LMEx (with and without filtering) and the combined extractions of LMEx with filtered dictionaries and RegEx. 99
4.7 The number of labeled candidates by a SME and the precision of labels (P-Valid and P-Invalid). 100
4.8 The precision (P), Recall (R), and F1 of the Micro- and Macro-averaged candidate entity classifier scores. 101
4.9 Pipeline performance after using the classifier on top of LMEx and RegEx methods, and with partial labels on the test corpora (C# from 1 to 10). Absent partial labels are replaced with strikethrough text. 102
4.10 Multi-domain NER performance comparison (Corpus # and precision/recall/F1 score). Gold data are labeled with IOB formatted labels. A SME-in-the-loop for KnowEx provides minimal supervision. 104

Acknowledgment

Prophet Muhammad (Peace be upon him) once said: “Whoever does not thank people (for their favor) has not thanked Allah (properly)”. [Musnad Ahmad, Sunan At-Tirmidhî]

I want to express my sincere gratitude to my advisor Dr. Krishnaprasad Thirunarayan (TK. Prasad) and Dr. Valerie L. Shalin for their continuous support of my Ph.D. research and work, the mentoring, motivation, and the relationship we formed throughout the years of my Ph.D. Wednesdays will always have a special meaning to me, the time we meet every week. Thank you for all the long hours of reviewing, discussing, and brainstorming together.

I would like to also thank the rest of my thesis committee: Dr. Keke Chen (Kno.e.sis, Wright State University), Dr. Guozhu Dong (Kno.e.sis, Wright State University), Dr. Steven Gustafson (noonum Inc.), Dr. Srinivasan Parthasarathy (The Ohio State University), for their insightful comments and encouragements, and for giving the time and effort to review my work.

My sincere thanks also go to The Ohio Center of Excellence in Knowledge-enabled Computing (Kno.e.sis), Wright State University, and their faculty and staff for the opportunity I had during my Ph.D. studies and research.

Thanks also to the National Science Foundation (NSF), the National Institutes of Health (NIH), the Computing Community Consortium (CCC), and Maana Inc for their funding and support of my studies and attendance to conferences during my Ph.D.

During my Ph.D., I was very fortunate to have worked with so many talented and dedicated people from whom I learned a lot. I thank all of my colleagues and collaborators for all the feedback, fun, and hard work we did together especially Patrice Chataigner (The Operations Partnership), Monireh Ebrahimi (Kansas State University), Shruti Kar (Wright State University), Jiayong Liang (The Ohio State University), Jiongqian Liang (The Ohio State University), Pranav Maneriker (The Ohio State University), Ewan Oglethorpe (Data

Friendly Space), Joy Prakash Sain (Wright State University), Nikhita Vedula (The Ohio State University), and Amir Yazdavar (Kansas State University). I cannot also thank enough my good friend Jeremy Brunn for all the help I got from him, and the fun times we had together all these years. Thank you, Jeremy, you were the best neighbor. Thanks also to John Aguilar, Tonya Davis, Alan Felming, Dipesh Kadariya, Austin Kemptonm, Kirill Kultinov, Jennifer M. Limoli, Mike Partin, Alan Smith.

I cannot thank enough Alexander Elkholy, Dr. Steven Gustafson, Dr. Subu Kandaswamy, Jason Mackay, and the rest of the team at Maana Inc. for the opportunity I had during summer 2017 as a research intern. It was an enriching opportunity I had by being part of the very energetic team which lead us to a product, a publication, and a patent by the end of my internship. Also, part of my work there became part of this dissertation.

I am also grateful to Jessica Glover, Lyle Schofield, Patrice Seyed, Amy Sheide, and the rest of the NLP team at 3M - Health Information Systems, MD, for the opportunity I had there during the summer of 2018 as an NLP intern.

Before, during, and after my masters' studies, Dr. Mansoor Alam, Dr. Henry Ledgard, and Dr. Gursel Serpen of the University of Toledo have always supported me, my education, and my Ph.D. application even outside UT. Their words of support and encouragement cannot be matched with words to express my gratitude. Thank you very much!

I would also like to thank my brother Dr. Khalid Al-Olimat and his wife, Dr. Feng Jao, for their support and for always being there for my family and me. Thanks also to my brothers and sisters for their continuous encouragement and support as far back as I can remember: Saad, Yaseen, Amira (Abeer), Abdullah, Fadwa, Dr. Ali, Dr. Muhammad, and Eng. Safa. Thanks to my mother for her patience all these years and for teaching us about the value of education. To my late Dad, thank you for being the role model that always reminded me about what really matters. I hope I can live up to your expectations. I want to thank my wife, Eng. Maram Abdelhadi for her unconditional support and limitless patience. To my beautiful daughter Salma, thanks for being there to push me to finish my

dissertation so I can have more time to enjoy your company. I say to all of you, I can't reward you, but I ask Allah (God) to reward you all in the hereafter.

Finally, I am really delighted and excited to have finished my terminal degree in the field I love. For the people who have helped me in various stages of this journey and in achieving this milestone and taught me so much, I want to thank all of you, the teachers, professors, professionals, bloggers, and coders. Special thanks also to Wikipedia, Coursera, edX, Udacity, Github, Lucidcharts, JetBrains, and the open-source community. Thanks to all the positive, kind, awesome, and inspiring people everywhere.

إِنَّ صَلَاتِي وَنُسُكِي وَمَحْيَايَ وَمَمَاتِي لِلَّهِ رَبِّ الْعَالَمِينَ

My prayers and sacrifice, my life and death, are all for God, Lord of all the Worlds

Introduction

1.1 Overview

Information Extraction (IE) techniques recognize entities and extract relationships and other detailed information from unstructured text. In this dissertation, I focus on the task of *entity extraction*, i.e., Named Entity Recognition (NER), which consists of multiple sub-tasks. *Entity delimitation* is typically the first step of entity extraction, which is meant to identify the boundaries of an entity mention in the text. Depending on whether we are extracting multiple entity types simultaneously, our NER techniques should be able to type the delimited entities with the correct class (e.g., Location, Organization, or Auto Part Name). The final step to NER is usually the linking step, which also includes resolution or disambiguation. This step links the entity mention of a particular class to its corresponding unambiguous record in a knowledge base or knowledge graph (e.g., linking the location name "Dayton" to its dictionary (aka., gazetteer) record, since gazetteers contain location names and all of their spatial information).

The challenges posed to entity extraction techniques differ between domains due to differences in sentence structure and delimitation issues. The majority of the techniques in the literature focus on designing supervised machine learning methods [135], which are not very practical due to the high cost of obtaining annotations and the difficulty in creating high quality (in terms of reliability and coverage) gold standards. Therefore, semi-supervised and distantly-supervised techniques are getting more traction lately as they tend

to bootstrap learning [118].

Top-down (knowledge-driven) and bottom-up (data-driven) NER approaches rely on internal and external evidence information to extract named entities from unstructured text [84]. The internal evidence is the one we gather from the text itself (the sequence of characters); from the surface form of the tokens. On the other hand, we gather external evidence from the contextual features of the mention in the text, such as the bag of surrounding words. Due to the differences in business contexts, enterprise data sometimes would have a different definition for some entity types, such as a product name [18], making the extraction task using only the internal evidence a challenging one. Additionally, while rule-based systems are widely used in industry [18], they still capture a specific need for a particular application. Rule-based systems achieve high precision but still suffer from low recall due to the *sparsity and specificity of rules*. Similarly, the use of supervised models, which usually exploit internal and external evidence, is unattractive due to the high annotation cost. New labels are usually required each time we have a different use case due to *data sparsity* and *semantic drift* problems, such as what a *product* is for a given company (e.g., "Chrome" being a product of Google, which is different than the common chromium surface treatment to metals) [126].

The challenges faced by unstructured social media data are of no less import. The ungrammatical, ill-formed, and improper sentence structure exhibited by social media content requires creative IE techniques to interpret them because natural language syntactic or semantic cues may be inadequate to rely on. The novel use of social media data, e.g., to monitor a real-time event, also poses constraints on the type of technique that can be used[1]. NER from text streams with novel domains, novel linguistic contexts, and different language styles across countries make deep neural networks or any supervised technique, which requires a considerable amount of data or extended time for training, impractical.

While there are technical challenges posed on the delimitation problem, such as the

[1] See our disaster response use cases of location extraction from targeted text streams in [54, 76]

ones mentioned above, other non-technical or user-centered design-aware challenges might also arise. These challenges would influence the design of the delimitation and linking techniques, such as when trying to accommodate practical and intellectual property concerns (e.g., ease of use and privacy preservation).

In my opinion, all of the following are key in helping to tackle the challenges mentioned above: the use of background knowledge, the monitoring of NER models certainty (while exploiting a model's encoded knowledge), and the full exploitation of inputs from a subject matter expert (SME)-in-the-loop. In this dissertation, I focus on lowering the cost of building NER pipelines with the help of available knowledge without compromising the quality of extraction while simultaneously taking into consideration feasibility and other concerns such as user-experience. I explore a type of distantly supervised (dictionary-based), supervised (with reduced cost using entity set expansion and active learning), and minimally-supervised NER approaches. Below, I discuss the various aspects of the knowledge-enabled NER approaches in this dissertation and how and why they are a better fit for today's real-world NER pipelines in dealing with and partially overcoming the difficulties mentioned above.

1.1.1 Knowledge Sources, Creation, and Reuse

Knowledge is equivalent to expertise and required to perform complex tasks [89]. It is also known that knowledge has two dimensions—the knowledge in people's heads (i.e., tacit or hidden) and the codified and captured/elicited knowledge (i.e., explicit) [82]. Rules can capture and represent the SMEs' tacit knowledge [129], their expertise in their field, and their judgments given a particular task and in specific situations. Additionally, the interaction between humans and machines improves the usefulness of machines and their models as they benefit from the knowledge transfer process [21].

Knowledge is either reused or created [82] and is captured using logic (as in symbolic AI) [95] or other forms such as in machine learning models, informal knowledge bases, or

databases. In this dissertation, I use readily available conceptual knowledge, the knowledge that was created by others at some point and captured in any form— dictionaries, taxonomies, knowledge graphs, pre-trained machine learning models, gold labels, or rules (the most commonly used form for knowledge representation [95]).

Knowledge creation, on the other hand, is what we do by capturing the SME's input (the expert's knowledge elicited/captured from their inputs or feedback by being part of the pipeline as human-in-the-loop), by training machine learning models from available/created information and knowledge (i.e., inductive learning), by combining knowledge from different sources, and by deriving knowledge from information and knowledge sources (such as creating n-gram models from dictionaries). In inductive learning, the knowledge is elicited/captured from negative and positive labeled instances (e.g., valid and invalid instances of an entity class such as Disease Name). Knowledge can also be derived from data using analogical learning techniques [95], such as Entity Set Expansion (see Chapter 3). Entity Set Expansion involves learning through the use of similar concepts and solutions. For example, the technique retrieves candidate entities by analogy through the use of textual information (i.e., explicit linguistic features) and explicit knowledge (from online sources such as WordNet [88]).

The intuition behind using background knowledge and a human-in-the-loop is that if knowledge can improve and affect the human's comprehension of scientific texts, it should also improve the performance of NLP models when used effectively [56]. Ultimately, this aims to improve the performance and accuracy of systems, and to reduce the cost of building them. Furthermore, as the elicitation of new knowledge from SMEs is slow, tedious, expensive, and error-prone [95], a reduction in the number of labels or requiring different forms of labels from SMEs might be wise (as I will show in Chapter 3 and 4).

In reality, the knowledge used to solve complex real-world problems can sometimes be uncertain or imprecise (i.e., contain errors and noise or is vaguely defined), incomplete (i.e., missing in part or whole), and sometimes inconsistent (e.g., in the case of multiple knowl-

edge sources) [95]. In the context of NER, uncertainty and imprecision can be introduced when the SME-in-the-loop provides wrong inputs or wrong labels, or when a dictionary contains noise and wrong values (such as wrong examples of class membership). These issues can also be introduced when there are disagreements between human annotators, when the task is vaguely defined or when it comes to subjective judgements. Hence, knowledge acquisition is very difficult and challenged by the availability of resources, the acquisition techniques, and the knowledge quality requirements. We avoid the knowledge acquisition bottleneck by using off-the-shelf dictionaries, any available background knowledge, and diverse knowledge sources. We also automate knowledge acquisition using machine learning from other sources of knowledge or different information sources (such as learning sequence models from labels and word embeddings from the unstructured text). As for the errors introduced from human labelers, I did not study that effect in this dissertation. Instead, I emulated the SME input using gold labels from pre-annotated gold standard datasets available online (such as CoNLL-2003 [127] or GENIA [99]).

Verification procedures done for checking the consistency and completeness of knowledge can be done for rule- and logic-based systems. However, as we are not building logic rules, it is not very easy to test the correctness and consistency of our knowledge. Hence, we rely on probabilistic models that we learn from gold labels with the help of the knowledge to overcome the negative effect of noise. Additionally, we design a few rules to filter and augment dictionaries in order to lower the effect of noise and incompleteness (see Chapter 2).

1.1.2 NER from Text Streams using Dictionaries

Targeted text streams collected using a set of keywords or a bounding box from sources such as Twitter are of high value and are very informative [64], and have the potential to satisfy an event-related information need [103], especially in the case of natural disasters [91]. However, for a given piece of information to be actionable, it should be pinned to

a point on the map, enabling context-aware computing [50, 66]. For example, unless we know where **Ganapathy Colony** in the tweet "water level in Ganapathy Colony is around 2 m" is, we will not be able to use the water level information in a storm surge model or in models where sensor readings are missing.

The mention of an entity, such as a location name, regardless of the linguistic context, can be extracted/matched using the lexical surface form (i.e., the sequence of characters) of that mention when matched with a dictionary record. Hence, the use of a dictionary to aid in the delimitation task is an excellent alternative to supervised techniques, especially in the case of stream processing, where rapidly evolving linguistic contexts would be a challenge to cover adequately. Having said that, the design choice of such context-agnostic extraction methods still poses a challenge when faced with semantic drift and context-sensitive mentions [126]. However, as I will discuss in Chapters 2 and 4, the sole reliance on the internal evidence for extraction is highly dependent on the entity class for which we are trying to extract. Some entity classes, such as chemical names or location names, can be extracted with reasonably good precision as they are less prone to homonymy. On the other hand, other instances of entity classes such as restaurants names or anatomical organs, which are context-sensitive, require external evidence to decide whether a given mention is an instance of the entity class or not (e.g., *head* in "the *head* of the animal" vs. "the *head* of the department").

In Chapter 2, I introduce our location name extraction method (LNEx). The technique uses lexical statistics (using language models built from region-specific gazetteers) to spot location mentions in targeted Twitter text streams (collected using event-specific hashtags). LNEx extracts fine-grained location mentions in a stream from an area of interest with high accuracy, without relying on explicit and computationally complex syntactic and semantic analysis.

Comparison with other techniques

The distinction between delimitation and linking/resolution/disambiguation is important, particularly in the location extraction literature. To address the delimitation problem, other researchers [71, 75, 52] have applied both syntactic heuristics (using lexical cues, e.g., "*in* New Orleans") as well as semantic heuristics (i.e., content-based, for different types of locations such as *buildings* and *streets*). These heuristics have serious limitations, such as failing to delimit metonyms[2] and location names that begin sentences (i.e., outside locative expressions, e.g., "*New Orleans* is flooded"). They also cannot assist in hashtag segmentation (required for the extraction of locations from hashtags). Moreover, simply identifying a location name still leaves open the problem of linking the entity to a corresponding gazetteer record for geocoding, which is fundamental to subsequent actions. Simple fixed phrase matching with gazetteer entries, as in [86, 75], solves the linking problem but remains vulnerable in two respects. With simple fixed phrase matching, the tendency for authors to shorten names while the gazetteers extend names creates conflicting conditions causing poor recall. On the other hand, simply relaxing matching criteria exacerbates the disambiguation problem.

Lexical variations, and dictionary noise and incompleteness

Gazetteers are invaluable resources as they contain location names along with their associated spatial information (e.g., latitudes and longitudes). Having such gazetteers would allow us to link entities to their associated meta-data needed for the above mentioned context-aware computing. Our method (LNEx) treats location names as a sequence of ordered words known as *collocations* [80]. Collocations are neither strictly compositional nor always atomic. We cannot identify them with grammatical rules, and fixed phrase matching is not reliable for longer names. Fortunately, gazetteers provide a resource for

[2]Metonyms are words or expressions that are substitutes of other things, such as "Washington" being the US federal government.

establishing region-specific naming regularities. Given a region-specific gazetteer, which retains the same location/spatial-context as the text, we can construct a statistical model of the *token sequences* it contains. However, location gazetteers are overly specific in two respects. First, due to pragmatic influences on writing style for example in conversation-like social media, users tend to shorten names to reduce redundant content. We call this the *location name contraction problem*. For example, "Balalok School", appears in the Chennai flood tweets in contrast to the full gazetteer name, "Balalok Matriculation Higher Secondary School". This phenomenon, therefore, requires us to augment the gazetteers with different surface forms of some location names, to overcome the lexical variation between mentions in text and their respective records in a gazetteer (e.g., adding "Balalok School" as an alternative name for the long gazetteer record). Second, gazetteers are imperfect, including highly ambiguous, or adjacent auxiliary (and sometimes extraneous) content. For example, they sometimes contain personal names as location names, such as "George, Washington", that threaten precision. Extraneous lexical items attached to location mentions also threaten recall, such as bracketed usage, status, or branding (e.g., "Location Name (Private)" or "Location Name (New)").

Dictionary augmentation and filtering

Carroll [15] examined the complex phenomenon of alternate name forms (called Nameheads). The study distinguishes between four shortening processes: (1) *Appellation Formation*, (2) *Explicit Metonomy*, (3) *Category Ellipsis*, and (4) *Location Ellipsis*. Our method in Chapter 2 partitions the location NER into two tasks: *location delimitation* and *resolution/linking/disambiguation*. Carroll's first two subprocesses fall under disambiguation more than delimitation. Disambiguation requires situational context for name resolution, which is beyond the scope of this dissertation. I therefore only briefly describe *Appellation Formation* and *Explicit Metonomy* before discussing the *Category Ellipsis* and *Location Ellipsis* processes in detail.

Appellation Formation occurs when, for example, the author refers to the location name "The Erie Canal" as "The Canal". People may also refer to the only airport in a city as just "The Airport". Referring to "Cleveland Hilton" as "Cleveland" is an example of *Explicit Metonomy*, where "Cleveland" is a possible metonymic variant of "Cleveland Hilton" in a narrow situational context (e.g., when an employee works for the Hilton Corporation they might accept it as a variant of the original name) [15]. Common ground or shared understanding between the author and the recipient establishes the referent [108]. Such contractions require situational context such as the author's location to resolve. *Explicit Metonomy* is at least as context-sensitive as *Appellation Formation* and likely to depend on anaphora resolution for disambiguation, i.e., prior exchange concerning workplaces. Therefore, both pose *disambiguation* problems, and require situational context such as the author's location and other background information to resolve.

In contrast, *Category Ellipsis* and *Location Ellipsis* pose *delimitation* problems that can be resolved with a statistical language model. *Category ellipsis* occurs when the author strips words related to the location category (e.g., "City" from "Houston City" to become "Houston"). *Location Ellipsis* occurs when an author drops the specific location reference in the location name (e.g., when "New York Yankee Stadium" becomes "Yankee Stadium" or "Cars India - Adyar" becomes "Cars India").

What we observed in the OpenStreetMap gazetteer is inconsistency in representing category ellipsis and location ellipsis, and that would account for some of the mismatches between location mentions and gazetteer records. To mimic these processes, we judiciously apply a skip-gram method to token sequences in the gazetteers, thereby including, for example, "Balalok School" as a variant of the complete name and point to the same original gazetteer record. Second, we eliminate auxiliary or ambiguous gazetteers' content (e.g., "George, Washington") that would otherwise threaten recall. LNEx requires a bounding box of the area of interest to build initial region-specific gazetteer and then augment it automatically for locations delimitation. An added benefit to using region-specific gazetteers

is that they improve the disambiguation process as only the location inside the area of interest should be extracted. For example, despite the abundance of "Main Streets" all over the US, a region-specific gazetteer can be focused enough to include just one "Main Street".

In Chapter 2, I demonstrate the successful spotting of location names in targeted texts using lexical statistics (word and phrase frequencies) built from high-quality augmented and filtered gazetteers. Although LNEx relies on a targeted Twitter stream, it does not exploit the unreliable metadata attached to tweets (such as latitudes and longitudes), as they, in the majority of the cases, do not provide accurate spatial context for the text in tweets. Also, LNEx is well-suited for stream processing, easy to set up and use, does not require any supervision (i.e., training data), and needs only freely available data.

1.1.3 User-centered and Cost-effective NER

For a machine learning engineer or a data scientist, the easiest way is to use a NER pipeline that does not need labels, hand-crafted rules, or a human-in-the-loop (similar to the above dictionary-based approach). However, for an evolving domain, a domain with no available knowledge, gold data, rules, or previously trained models (needed for transfer learning), the only way is to label some data, build a knowledge base, write some rules, have a human-in-the-loop, or a combination of some or all of the above. However, due to the associated cost, it is desirable to build cost-effective and user-friendly NER pipelines that would take into consideration the number of labels required, the design of the labeling task, the computational cost associated with the designed technique, and the ways we can reduce and improve all of the above.

Many practical limitations put a ceiling on the success of NER techniques. Some of these techniques suffer from noisy or unavailable annotations (especially for domain-specific entities) due to the cost, lack of resources and expertise, the time it takes to set-up labeling guidelines [25], the risk of discarding crowd-sourced labels [44], or incomplete and weak hand-crafted rules and patterns [1].

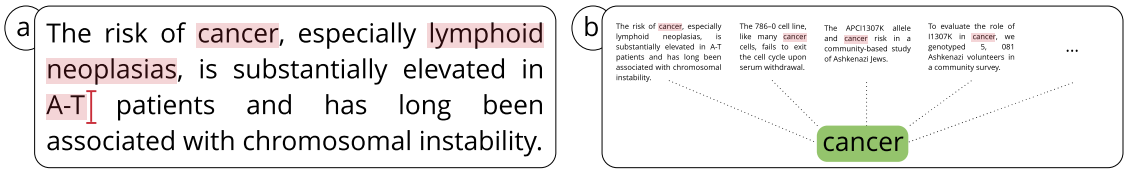


Figure 1.1: Two SME-in-the-loop labeling techniques: (a) disease mentions span labeling and (b) cancer concept binary labeling.

Hence, the annotation approach (affecting user experience) and training data (affecting cost) are the two focal points of any supervised method. In Chapters 3 and 4, I present our two NER techniques, which, with the help of a human-in-the-loop, knowledge, automatically induced rules, active learning, or transfer learning, can reduce the cost of building fairly accurate NER pipelines for different domains including medical, news, and user-generated content. Below, I introduce some of the key points regarding these systems.

User-centered design (UCD)

Some of the main UCD requirements in the context of NER concern data privacy, annotation cost, annotation approach, and feedback. Data privacy concerns arise from the fact that enterprises do not want to compromise their intellectual property (IP) by crowd-sourcing their labeling tasks. While this concern can be eliminated with in-house annotations, the high cost remains, in this case, requiring us to lower the cost of annotations to preserve privacy. Besides, the majority of the time, such labeling tasks require domain expertise and careful annotations, making crowd-sourcing difficult, and prone to duplication and many modifications.

Furthermore, mainstream entity annotation approaches typically present a large body of text or full documents to annotate entities, which can be time-consuming as well as lead to low-quality annotations (see (a) in Figure 1.1). Additionally, the majority of these techniques require labeling for multiple entity classes, which makes the annotation task harder and more complex. Therefore, sentence-level and single-entity-class annotation are

desirable to improve tractability and reduce cost, which also ensures a much improved human-in-the-loop experience. Moreover, using knowledge and transfer learning, minimal supervision (with reduced cost in terms of time and quantity), and minimized required user feedback (such as requiring binary labels instead of labeling text spans), can also improve the user experience (see (b) in Figure 1.1 where this context-agnostic labeling technique lowers the cost by not requiring the user to read the source sentences that provide these concepts are coming from).

The evaluation criterion is a point often overlooked as an important UCD requirement. NER pipelines, once designed, need some gold data to be evaluated on to test the quality of the built models. As we are usually building the pipelines for a novel domain, we will typically not have any gold labels to evaluate the system on. Therefore, an unsupervised evaluation technique is highly desirable in this case. Below, I will discuss how using pool-based active learning can help in this by providing a confidence measure on the subject matter expert (SME) labels.

NER cost

The cost associated with developing and running NER pipelines can be estimated based on many aspects: (1) the availability of pre-annotated data for building models, (2) the number of labeled sentences or entities required (the less, the better), (3) the need to modify and redo any of the annotations (e.g., in the case of misunderstanding of the labeling requirements), (4) the type of labels required from SMEs (e.g., binary labels being less costly than labeling of text spans in terms of time and effort), (5) the complexity of annotations (i.e., labeling for a single or multiple classes at the same time), and (6) the computational cost of building and running the pipelines.

There are many pre-annotated and publicly available corpora such as CoNLL-2003 [127], GENIA [99], and BioCreative II [33] that can be used to lower the cost of building NER pipelines. However, having such data leaves us with two challenges: (i) no train-

ing dataset can be found for many of the domain-specific entity classes in enterprise corpora due to data privacy concerns, IP restrictions, and problem-specific use cases (needing new/de novo annotations), and (ii) data sparsity, which hinders the performance of supervised models on unseen data (needing *incrementally augmented annotations* to the annotated sentence pool).

In Chapter 3, I present our active learning-based NER approach that reduces the number of data points required to only 55%. The approach achieves this desirable performance using a hybrid approach of entity set expansion (ESE) along with an entropy-based CRF sequence labeling approach allowing for active learning. While the ESE approach alone would allow us to sample the sentences with a higher potential to have an instance of the entity class for which we are labeling, the active learning approach allows us to rank order and then pick the sentences that are more likely to contribute to rapid learning (and therefore to remove any duplicate labeling). For instance, the combination of ESE and active learning contributed to a 45% reduction in the number of sentences, on average, required to achieve the same overall accuracy of the models if trained on all instances from the pools of sentences.

According to [34], the average reading rate of technical material is approximately 50 to 75 words a minute. By looking at ten datasets (see Section 4.5.1 in Chapter 4), the average number of tokens per dataset is around 112,910. So if we take the higher reading rate (i.e., 75 wpm), merely reading without labeling would require around 25 hours. Furthermore, ergonomic estimates of baseline error rates on such a detection-action task (i.e., labeling named entity spans) range from 0.011 to 0.16 *per word*, depending on the analysis method [41], assuming unlimited time and appropriate expertise. There is a dire need for improving such costly and error-prone task by minimizing text reading, avoiding span labeling, and reducing the complexity of labeling by annotating for a single class rather than multiple classes at the same time. However, imposing the UCD requirements above, including loosening the labeling requirements to binary labels, can cause what we

call a labeling starvation problem (i.e., querying annotators to label sentences containing a low frequency of entities from the desired class, see Chapter 3). Therefore, we designed the graph embedding-based ESE approach mentioned above to work side by side with active learning to reduce the labeling starvation problem (e.g., where some entity classes are present only in 8% of the sentences). ESE accelerates the learning rate by incorporating and exploiting the semantics of the desired entity class to more informatively sample sentences that are likely to have entities of the desired class. Ultimately, if each binary label takes, on average, a generous 10 seconds, the full annotations will take close to one-hour in comparison with 25 hours only for reading.

Moreover, as we aim to annotate all mentions of a single entity class in the corpora, our design choice lowers annotation cost (while relaxing the requirement of full coverage) by using a realistic approach of flexible stopping criteria using online evaluation. The online method calculates the confidence of a model on unseen data without the need for a held-out evaluation set. Having such an online feedback method supports incremental learning, allowing us to partially overcome the data sparsity problem (see Section 3.2). Also, it annotates a corpus without requiring annotators to scan all sentences using rapid auto-annotation based on predefined confidence levels (e.g., accepting the induced labels by a given model if the entropy of the labels is below 20%). In Chapter 3, I describe in more detail, the incremental learning solution (SpExtor) we developed for the extraction of sparse entities, and how we use ESE to accelerate the annotation.

Finally, the computational cost of building and executing these pipelines are of no less import. While recent NER models have achieved commendable accuracy improvements, they still require vast computational resources, which is not only expensive to build and use but also consumes a significant amount of energy. For example, training a BERT model would cost anywhere between $2,000 to more than $12,000 in cloud computing cost and close to the same CO_2 emission of an air trip for a single passenger from New York to San Francisco [123]. Luckily, the solutions introduced in this dissertation can be built and live

on a commodity laptop or server and would cost much less than the alternatives making it among the Green AI options [114].

1.1.4 From Candidate Entities To Named Entities

Given a sentence such as the following from the BioNLP'13 corpus [93]:

> c-myb is a frequent target of retroviral insertional mutagenesis in murine leukemia virus - induced myeloid leukemia.

A fluent English reader who has never before encountered the domain-specific terminologies, such as "c-myb", "murine leukemia virus", or "myeloid leukemia", would nevertheless infer that they are concepts of some kind. The reader would have reached this conclusion by examining the syntax and the morphological forms of these concepts. We try to extract all such concepts and label them as candidate entities. We featurize them to learn *what is*, and *what is not* an entity of the desired class by judiciously combining the lexical, syntactic, and semantic patterns characterizing them.

Our technique (KnowEx) extracts promising candidate entities, employs a SME-in-the-loop for labeling and filtering, and learns to classify the candidates into valid/invalid entities. The technique is minimally supervised, which exploits rich sets of multi-view features for candidate entities. The design of the method has key practical merit and is consistent with the rising interest in alternatives to fully supervised approaches that are costly to build, and are challenged by the requirement of scaling to multiple domains, and preserving IP and data privacy (see Chapter 2 and 3). KnowEx requires off-the-shelf domain-specific dictionaries, readily available online knowledge sources (e.g., WordNet, BabelNet, and pre-trained word embeddings), and minimal inputs from SMEs (i.e., binary labels). This allows us to achieve minimal supervision.

Candidate entities extraction

Using the same dictionary-based n-gram model from above, we determine the set of candidate entities that appear in all of the sentences of a given corpus using an off-the-shelf domain-specific dictionary. This language model-based technique allows us to determine the validity of an n-gram as a candidate entity in a sequence of tokens using a boolean function. Simple n-gram matching based on segment-based inverted indexing [65] or exhaustive matching [125] can replace this language model technique. However, these techniques are slower than our language model technique in Chapter 2 and our article [5]. For example, for the sentence from the BioNLP'13 corpus above, in a dictionary of organisms, "retroviral", "murine", "leukemia", and "virus" are valid unigrams but "c-myb", "mutagenesis", and the rest are not. Similarly, the trigram "murine leukemia virus", is a valid trigram matching a complete record from the dictionary, which makes it an organism candidate entity.

The incompleteness of dictionaries contributes to a reduction in the recall of the dictionary-based extraction techniques. However, regular expressions can expand the set of candidate entities from above [18]. We designed our second candidate entity extraction method (RegEx) to address this limitation. In our ESE method above, we use regular expressions to extract noun phrases as candidate entities and then ask the user to provide binary labels of yes/no as being an entity or not. However, those candidate entities were extracted using a single regular expression that did not scale well to multiple domains and suffered from low recall due to under-expressiveness [61]. We address this problem by automatically inferring regular expressions from a *set* of positive candidate entities exploiting the implicit sentence structure (i.e., using the descriptive tags of a sentence's tokens provided by part-of-speech (POS) tagging) then using these regular expressions for extraction. For example, in the LaptopReview dataset [104], the POS tags of the aspect terms "hard disk drive" and "1 GB of RAM" are $\langle$`JJ, NN, NN`$\rangle$ and $\langle$`CD, NN, IN, NNP`$\rangle$, respectively. Consequently, regular expressions can be constructed to extract such candidate entities using their

sequence of POS tags, as I discuss in Chapter 4.

Representation enhancement using knowledge

To learn what makes a candidate entity a valid instance of an entity class, we need rich feature sets and gold labels. Then, using these features and labels, we can learn binary classifiers to classify the candidate entities into valid/invalid instances. Available knowledge that we can exploit for the featurization of candidate entities allows us to enhance their representation, therefore allowing us to learn more accurate candidate entity classifiers. Hence, the use of knowledge allows us to minimize supervision by (1) constructing feature-rich embeddings of candidate entities for ranking and classification (allowing us to reduce the number of required labels for improving classification accuracy), (2) requesting SMEs to give only valid/invalid labels instead of labeling entity spans in text (reducing the effort of labeling). The use of pre-labeled instances of the other entity classes in the same corpora (as instances of the negative class), to automatically filter out the candidate entities set, can also improve precision, as I show in Chapter 4. For example, having gold annotations for some of the candidate entities labeling them as instances of some other entity class can be useful to remove them from the set of candidate entities for the current entity class for which we are labeling.

We create three feature views: (1) Lexical, Syntactic, and Semantic View, which exploits some unsupervised features from the text and the word senses of candidate entities from WordNet knowledge base [88], (2) Wikipedia Categories View, which exploits the Wikipedia categories retrieved using BabelNet [92] (such as "Rare_diseases", "Digestive_system", or "Urban_animals"), and (3) Contextual Embeddings View, which we construct using BERT contextual embeddings [27] as the bottom-up distributional semantics (an example of transfer learning).

Candidate entity classifiers can then be learned from these enhanced feature sets using binary labels from a SME-in-the-loop. Chapter 4 contains the details of the technique

and comprehensive empirical evaluation on public benchmarks spanning six datasets, to examine the scalability of the approach across multiple domains. The results show that the technique scales well to various domains, enhancing recall while maintaining precision in the face of noisy and incomplete off-the-shelf dictionaries.

1.2 Thesis Statement

> *Gathering explicit and contextual features for entity recognition is challenging due to data sparsity, the need for reliable annotation, and timeliness in the face of evolving data streams. Background knowledge, minimal supervision from subject-matter-experts, and machine learning models' certainty can be leveraged to develop reliable named entity recognizers at reduced cost.*

1.3 Research Questions

I base my thesis on the following research questions:

- **Question 1:** Can we accurately and rapidly spot named entities in text solely relying on a statistical language model synthesized from a dictionary or gazetteer? (Chapters 2 and 4, and in publication [5]).
- **Question 2:** How much improvement in accuracy can be gained by cleaning up the background knowledge used for NER? (Chapters 2 and 4, and in publication [5]).
- **Question 3:** How effective is the use of knowledge and smart sampling techniques in reducing the cost of building and running NER? (Chapters 3 and 4), and in publication [4]).
- **Question 4:** Can we provide an accurate online evaluation method that relies on

the certainty of a sequence labeling model to avoid creating a held-out set solely for evaluation? (Chapter 3 and in publication [4]).

- **Question 5:** What was the trade-off between the reduction in human annotation effort where SME's input is indispensable and the overall accuracy of the extraction method? (Chapter 4).

- **Question 6:** How effective and generalizable were the NER techniques presented in this dissertation in comparison with state-of-the-art and other similar techniques from literature? (In all chapters).

1.4 Contributions

In this dissertation, I make the following contributions:

1. I focus on the use of knowledge for named entity extraction problems, which lowers the cost and improves the human-in-the-loop experience by focusing on user-centered design requirements. I show that different levels of supervision can be achieved while using knowledge, which is key to different use cases such as stream processing. As a result, I provide different techniques developed with a focus on practicality, which lowers the cost, improves accuracy, and provides business value for industrial applications.

2. A method for preparing high-quality location name dictionaries (gazetteers) from online open data (e.g., OSM, Geonames, and DBpedia) and then deriving a language model from them for the task of location name extraction. I also provide a comprehensive analysis of the contribution of gazetteer quality to overall performance. Then, I demonstrate how our dictionary-based entity extraction (LNEx) convincingly outperforms commercial-grade NER and Twitter-specific tools with at least a 33%

improvement on average F-Score. Examples reveal the true challenges of location name extraction and the locus of tool failure in the face of these challenges.

3. A framework to annotate sparse entities rapidly in unstructured corpora using entity set expansion, active learning, and auto-annotation. The technique includes flexible stopping criteria using an online evaluation method without the need for gold-standard evaluation data, allowing us to learn sequence labeling models incrementally from annotated sentences without compromising the quality of the learned models. I also provide a comprehensive empirical evaluation of our sparse entity extraction framework (SpExtor) by testing it on six entity classes from three public datasets.

4. A competitive, knowledge-enabled, weakly supervised domain-adaptable named entity recognition technique (KnowEx). I provide a comprehensive empirical evaluation on public benchmarks spanning six datasets and ten entity classes, to examine the scalability of the approach across multiple domains. KnowEx exhibits inherent precision-recall trade-off and provides a general solution that scales to multiple domains using off-the-shelf dictionaries and generic knowledge with minimal supervision. The final results show that the pipeline improves recall by 27% while sacrificing 5% in precision, still improving the F1-score by 10% over the baseline.

5. A referent corpus that represents the full scope of location name extraction challenges and a challenge-based categorization of place names found in the corpora of three targeted streams. The corpus contains data from three different Twitter streams from flooding events in three different locations: the 2015 Chennai flood, the 2016 Louisiana flood, and the 2016 Houston flood, for the evaluation of our technique and also for use by others.

6. Open source implementations of the LNEx, SpExtor, and KnowEx frameworks. Codes and data can be found at https://github.com/halolimat/LNEx,

https://github.com/halolimat/SpExtor, and https://github.com/halolimat/KnowEx.

1.5 Dissertation Structure

Chapter 2 contains the details of the dictionary-based location name extraction method published in [5]. I give a detailed overview of the system's component, the data annotation criteria for evaluation, and illustrative examples comparing the performance of our method with ten other tools on the same task.

Chapter 3 contains the details of the active learning-based sparse entity incremental learning framework published in [4]. I give the details of the components and the methods used to achieve the goals of the system, including the information on the use of entity set expansion and graph embedding to enhance the learning speed.

Chapter 4 contains the details of the minimally-supervised NER technique. I describe the technique, how we extracted candidate entities, enhanced their features using knowledge and rules, and how we built the pipeline to allow for minimal supervision from SMEs.

Chapter 5 contains the conclusions of the dissertation and a discussion of possible future directions.

Location Name Extraction

Extracting named entities from informal and unstructured social media data requires the identification of referent boundaries and partitioning compound names. Variability, particularly *systematic* variability in named entities such as location names [15], challenges the identification task. Some of this variability can be anticipated as operations within a statistical language model, in this case, drawn from gazetteers such as OpenStreetMap (OSM), Geonames, and DBpedia. Language models permit evaluation of an observed n-gram in Twitter targeted text as a legitimate location name variant from the same location-context. Using n-gram statistics and location-related dictionaries, our Location Name Extraction tool (LNEx) handles abbreviations and automatically filters and augments the location names in gazetteers (handling name contractions and auxiliary contents) to help detect the boundaries of multi-word location names (aka., atomic toponyms [6]) and thereby delimit them in texts.

We evaluated our approach on 4,500 event-specific tweets from three targeted streams to compare the performance of LNEx against that of ten state-of-the-art taggers that rely on standard semantic, syntactic and/or orthographic features. Our technique improved the average F-Score by 33% to 179%, outperforming all taggers. Further, it is capable of stream processing.

2.1 Introduction

In this chapter, I introduce our LNEx technique [5]. The technique successfully spots location mentions in the text given a targeted Twitter stream (collected using event-specific hashtags) and lexical statistics (using language models built from region-specific gazetteers). LNEx extracts fine-grained location mentions in a stream from an area of interest, such as an event with a known zone or spatial extent, or a natural disaster with a known affected area. It extracts locations with high accuracy without relying on explicit and computationally complex syntactic and semantic analysis, such as analyzing the text for detecting locative expressions (as in "... across from LOC ...") or looking for semantic tags such as location categories in extracted candidate entities (as in "... LOC Street ...") [71, 75, 52].

Location names consist of a sequence of ordered words known as *collocations* [80], which reflect complex regularities implicit in the distribution of terms in the region-specific gazetteer that a language model can capture. Nevertheless, gazetteer matching is not a straightforward process due to two problems: (1) gazetteers contain extraneous information to location names (e.g., meta tags as "(Closed)" in "GM Locomotive (Closed)"), and (2) gazetteer records may differ from the references people use due to shortening and other processes that human written communication exhibit (e.g., referring to "New High School" as "New School"). These two challenges make gazetteers inadequate for direct phrase matching, which requires further augmentation and modification or filtering of the records. For filtering, we developed stopword lists containing phrases that we consider as extraneous or auxiliary, or ambiguous such as proper names (e.g., "George, Washington") that should be filtered out and removed. As for the augmentation, consistent with Carroll [15], we address two kinds of location name contractions: category ellipsis and location ellipsis. We augment gazetteers with skip-grams (e.g., to include "Balalok School" as a variant of the full location name in the gazetteer). The technique requires a bounding box of the area of interest to build initial region-specific gazetteer and then augment it automatically for locations delimitation. The data and the full source code of the technique are available

online at https://github.com/halolimat/LNEx.

In the remaining parts of this chapter, I demonstrate the successful spotting of location names in targeted text streams using lexical statistics (word and phrase frequencies) built from high-quality augmented gazetteers. Although our method (LNEx) works on a targeted Twitter stream collected using event-specific keywords, it does not rely on geo-coordinates associated with tweets, which are rarely available, and it does not need any supervision (i.e., training data). It is well-suited for stream processing and requires only freely available data. Our technique provides the foundation for localizing social media information on a map, thereby supporting demographic studies, disaster management applications, and other computational models. With the increased availability of open data, we expect our approach based on region-specific knowledge to be widely applicable in practice.

2.2 Challenges of Location Name Extraction

Table 2.1 contains the example challenges and different types of location mentions in the sample unstructured social media data that we examined. The following is a list of these types, highlighting the challenges of the location name extraction problem:

T1- Ambiguous locations: These are the locations that are context-dependent and cannot be resolved unless we have background knowledge about the author.

T2- Full locations in hashtags: These are typical locations that are written inside hashtags.

T3- Abbreviated locations: These include standard abbreviations of locations names such as “LA” for “Louisiana” and “OH” or “Ohio” and so on.

T4- Numeric locations: These include house numbers that map to a particular building in a certain street.

Table 2.1: Example challenges and types of location names in the social media text.

Now have a very nice lake in $\overbrace{\text{my backyard}}^{T1}$ accompanied with geese. #$\overbrace{\text{LouisianaFlood}}^{T2}$
$\overbrace{\text{Highway 288}}^{T6}$ in $\overbrace{\text{HOU}}^{T3}$ unflooded & back in business. Stay strong, $\overbrace{\text{HTown}}^{T11}$
2 ppl need to be evacuated frm # $\overbrace{\text{21}}^{T4}$, $\overbrace{\text{New Avadi Rd}}^{T5}$, $\overbrace{\text{Kilpauk Garden}}^{T2}$
$\overbrace{\text{LA 339}}^{T6}$ ($\overbrace{\text{Verot School Rd.}}^{T5}$) between $\overbrace{\text{US 90}}^{T6}$ and $\overbrace{\text{College Drive}}^{T7}$ in $\overbrace{\text{Lafayette Parish}}^{T7}$ # $\overbrace{\text{la}}^{T8}$ wx
Slow-moving storms from $\overbrace{\text{Central}}^{T9}$ to $\overbrace{\text{Clinton}}^{T10}$
$\overbrace{\underbrace{\text{sou th}}_{\text{T12}}\text{ kr koil street}}^{\text{T13}}$ near $\overbrace{\underbrace{\text{Oxford school}}_{T15}\text{ . }\underbrace{\text{west mambalam}}_{T13}}^{\text{T16}}$
urgent RT pls, any near $\overbrace{\text{saligramam}}^{T13}$ $\overbrace{\text{balalok school}}^{T14}$ elderly cpl, not reachable last 3 days

T5- Locations with abbreviations: These are the locations that contain an abbreviated location category such as “Ave” for “Avenue” and “Str” for “Street”. This type poses a lexical variation problem when matching location records from gazetteers and location mentions in the unstructured text.

T6- Alphanumeric locations: These location names can be very challenging to noun phrase-based named entity extraction techniques as they are not easily detected as noun phrases.

T7- Normal locations: These are the location names that can be found in a gazetteer and have none of the problems or challenges in the other types.

T8- Abbreviated locations in hashtags: These are the T3 locations in hashtags.

T9- Highly ambiguous locations-1: These include the common words used to name locations such as “Central” or “People” (a shop name in Chennai, India). A longer list of these ambiguous and very unusual names can also be found at en.wikipedia.org/wiki/Wikipedia:Unusual_place_names.

T10- Highly ambiguous locations-2: These are homographs that have the same spelling as location names, but they are not location names such as “Turkey” and “Clinton” or given names such as “Paris” and “Berlin”.

T11- Nicknamed locations: These are locations that have alternative nicknames such as “H-Town” for “Houston”, “Big Apple” for “New York”, or “Beantown” for “Boston”.

T12- Misspelled locations: Sometimes, location names, like any other token in a sentence, can be misspelled. Later in the chapter, I report our findings on misspelled locations, which shows that they are not statistically significant.

T13- Wrong cased locations: These are locations that were mistakenly lower cased w.r.t the first letters, which would result in challenging part-of-speech (POS) tagging.

T14- Contracted locations: These are locations that authors contracted into fewer tokens than the full location names by dropping the more general terms while retaining more specific ones. For example, "Amman High School" can be contracted to "Amman School" by retaining the essential tokens and ignoring the general tokens like "High", which can be part of many other location names.

T15- Mixed cased locations: Locations that have their case mixed, suggesting that the following tokens might not be part of the same phrase.

T16- Ungrammatical writing: This highlights ungrammatical and ill-formed sentences in social media data posing challenges to the tokenization and parsing procedures.

In the next sections, I explain our solution for extracting locations from tweets of the above types. Figure 2.7 also shows the prevalence of these types of location references in a random sample from our three datasets.

2.3 Approach

I discuss the details of our approach in four subsections. First, I present the general idea of statistical inference via n-gram models. The core of LNEx is a statistical language model consisting of a probability distribution over sequences of words (collocations) that represent location names in preexisting, region-specific gazetteers. Then, I separately discuss several modifications to both gazetteers and text samples, including gazetteer augmentation and filtering, spelling correction, and hashtags breaking and tweet preprocessing. Finally, I

illustrate the full location analysis and matching process that reliably spots location names and their boundaries in text.

2.3.1 Statistical Inference via n-gram Models

LNEx constructs an n-gram model from the collocations that exist *in the gazetteer* to determine the valid location names (LNs) that might appear in tweets. Given tweet content such as"texas ave is closed", the model can check the validity of one to n-grams. From the gazetteer, "texas" and "ave" are valid gazetteer unigrams, but "is" and "closed" are not. Similarly, "texas ave" is a valid and preferred bigram (over two unigrams).

Specifically, as shown in Algorithm 1, we first tokenize all location names in the gazetteer to construct the n-gram model and then save the resulting lists of unigrams, bigrams, and trigrams (Lines 2-5). Next, for bigrams and trigrams, in Lines 6-9 we create Conditional Frequency Distributions (CFD) to count the collocations (i.e., $c(\cdot)$ in Equations 2.1-2.2). Conditional Probability Distributions (CPD) are then constructed from the recorded n-grams using Maximum Likelihood Estimation (MLE). We found in our dataset that roughly 98% of location mentions in tweets have less than three words. Therefore, we assume that only the previous two words determine the probability of the next word (Markovian assumption of order two). MLE assumes zero probability values for tokens missing from the gazetteers. *This data sparsity problem is mitigated by augmenting the gazetteers with location name variants* (see Section 2.3.2). In lines 11-13, we determine the validity of an n-gram (the string s) using the boolean function `VALID-N-GRAM(s)` with the help of Equations 2.1-2.4, where $c(w_x^y) \equiv c(w_x w_{x+1} \dots w_y)$, w_x^y is the collocation count (i.e., the occurrences of consecutive words, w_x to w_y), $P(w_z|w_x^y)$ is the conditional probability of a word w_z given previous collocation w_x^y, and the chain of probabilities P_1 (for unigrams), P_2 (for bigrams), and P_3 (for tri or larger grams).

Algorithm 1: Language model generation algorithm using terms distribution in a region-specific gazetteer. We create conditional frequency distributions (CFD) to count location names, then calculate the conditional probability distributions (CPD) using maximum likelihood estimation (MLE).

Function `Compute-Model(`$Gazetteer$`)`**:**
    for $ln \in Gazetteer$ **do**
        $unigrams \leftarrow tokenize(ln)$
        $bigrams, trigrams \leftarrow$ generate from $unigrams$
    end
    for $n\text{-}grams \in [bigrams, trigrams]$ **do**
        CFD $\leftarrow$ create using $n\text{-}grams$
        CPD $\leftarrow$ create using ***CFD***
    end

Function `Valid-N-Gram(`$string = s$`)`**:**
    $w_1^n = (w_1, \ldots, w_n) \leftarrow tokenize(s)$
    return $P(w_1^n) > 0$ ▷ calculated using the Equations 2.1-2.4

$$P(w_z|w_x^y) = \frac{c(w_x^z)}{c(w_x^y)} \tag{2.1}$$

$$P_1 = P(w_1^1) = \frac{c(w_1)}{\sum_{i=1}^{|\text{unigrams}|} c(w_i)} \tag{2.2}$$

$$P_2 = P(w_1^2) = P_1 \times p(w_2 \mid w_1^1) \tag{2.3}$$

$$P(w_1^n) = P_2 \times \prod_{i=3}^{n} P(w_i \mid w_{i-2}^{i-1}), n \geq 3 \tag{2.4}$$

2.3.2 Gazetteer Augmentation and Filtering

We faced two primary challenges when building our language model using raw gazetteers, which are not adequately explored in previous work [86, 132]:

1. **Conditional Collocation Contractions**: Some atomic n-gram location names (collocations) cannot be shortened, e.g., "New York". However, contraction can preserve the meaning of longer names, especially when the first and the last words denote a specific part and a generic part, respectively, such as in "Balalok School".

2. **Auxiliary and Spurious Content**: Gazetteer entries may contain extraneous content that can cause location matching to fail. Consider the examples in Table 2.3, where *bracketed strings* can refer to the usage, status, or branding. *Hyphens* also are used to reflect a part-of relationship between two location names, a relative reference to another location name, or the names of two places connected by a road. Cleaning such entries improves matching reliability.

To address these challenges and inform a generative method for gazetteer augmentation and filtering, we implemented variants of the two Nameheads subprocesses [15]: Category

Table 2.2: Gazetteer augmentation to generate alternative names.

Balalok Matriculation Higher Secondary School
Balalok School,
Balalok Matriculation School,
Balalok Higher School,
Balalok Secondary School,
Balalok Matriculation Higher School,
Balalok Higher Secondary School

Ellipsis (for collocation contraction) and Location Ellipsis (for filtering the auxiliary content). Category and Location Ellipsis can be readily used to augment gazetteers, allowing us to overcome much of the lexical variation between the authors' text and the gazetteers' records. *Category ellipsis* occurs when the author strips words related to the location category (e.g., any of the intermediate tokens inside "Balalok Matriculation Higher Secondary School" or "City" from "Houston City" to become "Houston"). *Location Ellipsis* occurs when an author drops the specific location reference in the location name (e.g., when "New York Yankee Stadium" becomes "Yankee Stadium" or "Cars India - Adyar" becomes "Cars India"). The following are the details of our implementation:

1. **Skip-grams**: Given a location name $t_1 \dots t_n$, we retain t_1 and t_n while varying $t_2 \dots t_{n-1}$. To avoid adding "City York" as a legitimate variant of the location name "City College of New York", we require t_n to be a location category name (e.g., building, road). Therefore, "Balalok Matriculation Higher Secondary School" generates {`Balalok School, Balalok Secondary School, ...`}, see Table 2.2. This technique results in a small number of contractions that are either useful collocations or are too random to cause many false positives [46]. For example, "New University" is mistakenly established as a legitimate variant of "New York University". The problem here is that "New York" is an inseparable, idiomatic listeme. A more sophis-

Table 2.3: Extraneous text in raw gazetteers.

	Content Description	Example Gazetteer Record
1	Descriptive Tags	(Private Road)
2	Life-cycle/Status Tags	Little Rock School (historical)
3	Alternative/Old Names	Scenic Road (Frontage Road)
4	Acronyms	International House of Pancakes (IHOP)
5	Hyphenations	Cars India - Adyar Pilot - Hammond Beacon Health - Westchase
6	Branding/Descriptive	TAJ KAZURA (A Comfort Stay)

ticated name model that ignores the generic parts and retains the specific parts when augmenting a location name (e.g., adding "Sam's" as a variant of "Sam's Club") is beyond the scope of the current work [24].

2. **Filtering**: We compiled a generic list of phrases to address bracketed auxiliary content and remove specific words on a case-by-case basis (e.g., 1-2 in Table 2.3). The remaining bracketed names are deemed legitimate alternatives (e.g., 3-4 in Table 2.3). We treat hyphenated location names as Location Ellipsis and split them on the hyphen and add the two splits (e.g., 5 in Table 2.3). We expect that the majority of these location names represent a partonomy relationship where the hyphen may be read as a "part of" relation between split tokens. We do not add the second token as a variant when it already exists in the gazetteer on its own as location name entity (e.g., "Hammond" in "Pilot - Hammond").

These two methods augment and filter partial OSM, Geonames, and DBpedia gazetteers sliced from the original sources using a bounding box for the area of interest. Further, we can attach the metadata of the original location name to the generated variants. Moreover,

Table 2.4: Statistics of location mentions inside hashtags, where, **# UHs** is the number of unique hashtags, **UHs*** are the unique hashtags excluding the ones used for crawling.

Dataset	**# UHs**	**% in UHs**	**% in UHs***	**% in all hashtags**
Chennai	229	37%	17.4%	26.3%
Louisiana	471	24%	17.3%	31.3%
Houston	274	29%	17.6%	28.3%

by treating derived names as synonyms for existing names, we avoid creating additional demands on disambiguation or equivalencing. We add the derived, variant location name to the gazetteer as long as it does not collide with an existing location name. Additional filtering of proposed variants is required to prevent false alarms. Similar to the use case in [132, 38], we compiled a list of 11,203 words, including 678 inseparable bigrams, such as "Building A", as gazetteer stop words. This list also includes unusual location names (e.g., "Boring" in Maryland and "Why" in Arizona) and proper nouns (e.g., "James" in Mississippi) that could appear as non-location tokens. We then eliminate from all gazetteers the location names that overlap with our gazetteer stop words to reduce false positives. Next, I discuss the modifications to text before the step of extracting locations.

2.3.3 Tweet Preprocessing

To complement the gazetteer preprocessing, we also require potentially non-trivial tweet preprocessing. We start by removing the retweet handles, URLs, non-ASCII characters, and all user mentions. Then, we tokenize tweets using TweetMotif's Twokenizer [98], which treats hashtags, mentions, and emoticons as a single token. We do not tokenize on periods (e.g., "U.S."). The use of Twokenizer improved the location extraction performance by around 2% F-score in the observed datasets.

Hashtag Segmentation: In our Twitter datasets, on average, around 29% of the hashtags include location names. Excluding hashtags used to crawl the data, approximately 17% of the unique hashtags contain locations. As the number of locations in hashtags is significant, similar to [75], we adopted a statistical word segmentation algorithm to break hashtags to their respective discrete words for location spotting [97]. We used the implementation provided in wordsegment module[1].

The algorithm uses a list of 333,333 types (distinct unigrams) and their counts extracted from Google's 1 billion token corpus. For a given hashtag (e.g., "#ChennaiFloods"), we remove the hash symbol and try to segment the hashtag to "chennai" and "floods". The method assumes independence between hashtag tokens and finds the best segmentation of a given string by maximizing its probability: $best = argmax_{\ c \in\ candidates(s)}\ P(c)$, where c is the current segmentation of the string s (e.g., $seg_1 =$ "ch" and $seg_2 =$ "ennaifloods"), $candidates(s)$ is the set of segmentations of the string s, and $P(c) = \prod_i P(seg_i)$.

Spelling Correction: We consider a tweet token as misspelled if it is an out-of-vocabulary token, where the vocabulary is gazetteer words and a large English vocabulary word list[2]. LNEx corrects all misspelled tokens using the Symmetric Delete Spelling Correction algorithm (SymSpell)[3] that is six orders of magnitude faster than Norvig's spelling corrector [97], which was used by [40] in their location extraction tool.

SymSpell is based on the Damerau-Levenshtein distance algorithm and finds matches for a given word from the dictionary of unigrams with the smallest edit distance. As computing edit distances between the query word and each unigram is expensive, SymSpell pre-generates all terms with an edit distance of ≤ 3 from each of the gazetteer unigrams and adds them to a dictionary during initialization. For example, $delete(new, 1) = \{ew, nw, ne\}$, where 1 is the edit distance. At the time of correction, SymSpell also generates all edit variations of the tweet token to match them with the previously generated

[1] http://www.grantjenks.com/docs/wordsegment/
[2] https://github.com/norrissoftware/words3
[3] https://github.com/wolfgarbe/symspell

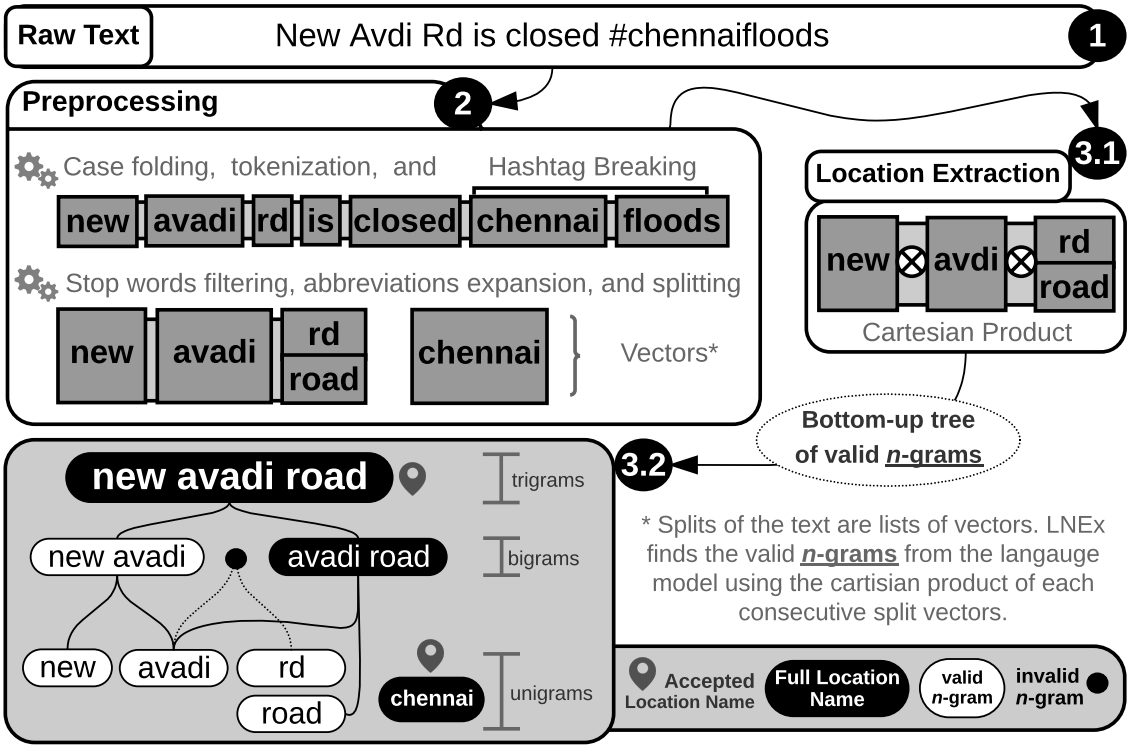


Figure 2.1: Extracting locations using the LNEx tool.

variations in the dictionary. As we shall see, spelling correction has only a small influence on system accuracy.

2.3.4 Extracting Location Names using LNEx

After modifications to the gazetteers and texts, LNEx extracts location names, as illustrated in Figure 2.1. In ⓐ, LNEx reads the raw tweet text, preprocesses it (as in Section 2.3.3) starting with case-folding. After tokenizing the tweet, the hashtag segmenter breaks hashtags into tokens. Later, stop words are used to split a tweet into consecutive word fragments where each tweet split of size n can have zero to n potential location names. We custom build the tweet stop list starting with around 890 words[4] excluding the gazetteer unigrams.

LNEx now takes each tweet split and converts each of its tokens into a vector of to-

[4] http://www.ranks.nl/stopwords

kens v using two dictionaries: the USPS street suffixes dictionary[5] and the English OSM abbreviations dictionary[6]. This conversion adds possible expansions and abbreviations of a token (e.g., "Rd" to "Road", and vice versa). Ultimately, this overcomes the lexical variations between location mentions in tweets and their corresponding gazetteer entries for robust matching.

In ⓑ, the language model is used to find the valid n-grams from the Cartesian product of the consecutive vectors. It builds a bottom-up tree for each tweet split starting from 1 to n-grams by gluing the consecutive tokens together if they represent a valid segment in the gazetteer. We improve the speed of the algorithm significantly by splitting the tweet and eliminating invalid n-grams. LNEx then selects a subset of valid n-grams from the tree; for overlapping n-grams, we prefer the longest full mentions (e.g., "New Avadi Road" over "Avadi Road") and keep both if they are of the same length. When full location names appear inside partial ones, we keep only full names in the set of possible locations (e.g., extracting "Louisiana" from "The Louisiana"). The final list of full location mentions found in all splits of a tweet comprises the result of LNEx.

2.3.5 Time and Space Complexity

LNEx extracts and links a full location mention to its corresponding gazetteer entry through a simple dictionary lookup that takes constant time $\mathcal{O}(1)$. The location extraction time is bounded by the time for creating the bottom-up tree of tokens which takes $\mathcal{O}(|v|^s)$ where $|v|$ is the length of the longest vector of token synonyms (i.e., all the expansions and abbreviations of a token) and s is the largest number of tokens with synonyms in a location name. Splitting the tweet into smaller fragments significantly lowers the asymptotic growth of the algorithm, enabling stream processing. In practice, for our dictionaries and gazetteers, $|v| \leq 4$ and $s \leq 3$. So, a pessimistic upper bound on the number of candidates for each

[5] http://pe.usps.gov/text/pub28/28apc_002.htm
[6] wiki.openstreetmap.org/wiki/Name_finder:Abbreviations

location (though rarely realized) is 4^3. The space complexity of the method is bounded by the product of the number of gazetteer entries, L, and the number of variants of a location name (Skip-gram method 1), that is, 2^{m-2}, where m is the number of tokens in a location name. Effectively, the space complexity is $\mathcal{O}(L.2^{m-2})$ where typically, $2 \leq m \leq 5$. Further, according to our tests, LNEx needed only up to 650 MB of memory and can process, on average, 200 tweets per second.

2.4 Evaluation

To demonstrate the effectiveness of our context-aware location extractor LNEx, we collected event-specific tweets from three different targeted streams corresponding to floods in Chennai, Louisiana, and Houston. Below are the details of how we collected, filtered (to get a set of tweets with higher potential to have location mentions in them), and categorized and annotated them (to benchmark each component of LNEx, and for comparing LNEx with other state-of-the-art tools for the location extraction task). Our approach requires an innovation in the conceptualization of effective performance.

2.4.1 Data Collection

To collect geographically limited, disaster-related tweets, we compiled a list of event-specific hashtags in order to filter the Twitter stream at the time of the event. The following is the list of hashtags used to collect each dataset:

- **2015 Chennai flood:** #chennairescue, #chennairainshelp, #chennaifloods, #helpchennai, #chennaifloodrelief, #chennairains, #chennaimicro, #chennaivolunteer, #chennaiflood, #volunteerforchennai, #tnflood, #chennaiweather, #tnfloods, #chennaiupdates, #chennaihelp
- **2016 Louisiana flood:** #louisiana, #lousianaflood, #lawx, #lafloods, #laflooding

- **2016 Houston flood:** #prayforHouston, #houwx, #houstonflood, #txwx

2.4.2 Data Filtering and Preparation

To retrieve tweets with higher potential to have location mentions in them, we sampled tweets from each dataset that contains one or more of the following terms:

- **Location verbs:** We consulted Levin verbs [62] to compiled a list of verbs that corresponds to a change of state which in many cases would accompany a location mention in a sentence. The list includes: avoid, cross, depart, escape, evacuate, leave, and shelter.

- **Building suffixes:** We took this set, which contains the suffixes of building names such as abbey, academy, airport, clinic, school, station, and university from [38].

- **Direction markers:** This set contains direction markers and their abbreviations, combination, and different forms such as north, south, south-west, nwest, and s-east.

- **Distance markers:** This set contains markers such as mile, kilometer and yard, and their abbreviations such as mi, km, and yd or yds.

- **Prepositions:** This set contains prepositions that accompany locations in sentences such as above, across, along, around, behind, beside, and outside.

- **Street suffixes:** We used the same dictionaries mentioned before, i.e., the USPS street suffixes and the English OSM abbreviations dictionaries to compile a list of terms and their abbreviations such as avenue, ave, bridge, brg, highway, hwy, parkway, street, str, and way.

After prepossessing the tweets as in Section 2.3.3, we lemmatize tweets using CoreNLP [81] and then use the list of location indicative terms from above to only keep the tweets that include these terms. Finally, we filtered out duplicates to retain unique tweets and sampled 1,500 from these filtered tweets for each dataset randomly.

2.4.3 Benchmarking and Annotations

Consistent with the problem of determining whether or not a location mention is inside the area of interest, our novel benchmark categorization scheme is based on where these locations lie in relation to the area of interest drawn using a bounding box. For example, with respect to Chicago, IL, USA:

1. in**LOC**: These are locations that are inside the area of interest, (e.g., Millennium Park or Burlington Ave.)

2. out**LOC**: These are locations that are outside the area of interest (e.g., Central Park, 5th Ave, New York.)

3. amb**LOC**: These are ambiguous locations that need context for identification (e.g., "our house" or "Louisiana Rivers".)

In contrast to [83, 38], our categorization is not based on location types (e.g., buildings, facilities, schools) but on the relative position (i.e., inLOC or outLOC) and the type of the location mention (i.e., inLOC or ambLOC). This approach identifies the true scope of the challenges in extracting location names. Other schemes that annotate for a limited set of location types, such as Geoparse Twitter Benchmark Dataset[7], miss other types of location mentions in tweets, such as "New Zealand" and "Christchurch", making the datasets annotated with this scheme incompatible for testing the tools in Section 2.4.8 including LNEx. On the other hand, our annotation scheme allows us to test any location extraction tool by ignoring the optional additional expressivity. Additionally, we do not make the distinction between whether a mention is a POI (hotel, restaurant) or a geo-location (country, city, river)— we consider all of them as locations.

Although disambiguation is out of the scope of this chapter, reducing the ambiguity of the extracted and linked location names is very desirable. Fortunately, only around 14%,

[7] https://web-001.ecs.soton.ac.uk/

Figure 2.2: Example location name annotations using the BRAT tool in the context of the 2016 Louisiana flood.

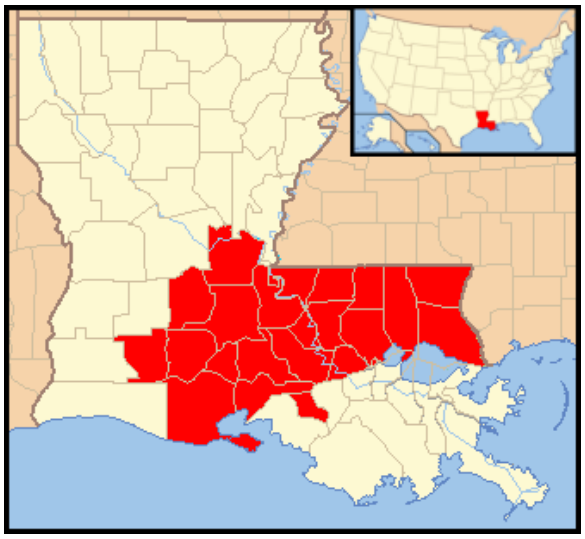


Figure 2.3: 2016 Louisiana floods map: the 21 Louisiana parishes that were declared federal disaster areas by FEMA (Source [130]).

24%, and 26% of Chennai, Louisiana, and Houston tweets respectively need disambiguation due to the linking of locations with more than one possible geo-coordinate.

Tweet Annotations Figure 2.2 shows an example of manual annotation from the Louisiana flood tweets using the BRAT tool [122]. It allows us to define search functionalities and additional resources such as Google Maps, Photon[8], and Mapzen[9] for the annotators to use.

In the rare case that no match is found, the annotators may consult other resources, including the standard Google Search. An ambLOC annotation covers cases not otherwise found. Annotators decide whether a location name is an inLOC or outLOC by consulting a pre-defined bounding box of the affected area as declared by officials (see the example map of the 2016 Louisiana floods in Figure 2.3).

[8]photon.komoot.de
[9]https://mapzen.com

We annotated three datasets: the 2015 Chennai flood, the 2016 Louisiana flood, and the 2016 Houston flood (collected with Twitter's streaming API using the most relevant hashtags of each event, e.g., #ChennaiFloods2015 and #LouisianaFloods2016). We recruited three annotators and had a meta annotator to review and resolve any issues in the data. In Chennai, there was 4,589 location names (75% inLOC, 4% outLOC, and 21% ambLOC); in Louisiana, 2,918 (66% inLOC, 13% outLOC, and 22% ambLOC); and in Houston, 4,177 (66% inLOC, 7% outLOC, and 27% ambLOC). We randomly selected 1,000 tweets (500 each from Chennai and Louisiana) as a development set and the remaining 3,500 as the test set for evaluation.

2.4.4 Evaluation Strategy

Because BRAT records the start and the end character offsets of the annotated LNs, we evaluate the extraction task by checking the character offsets of the spotted location name in comparison with the annotated data. We used the standard comparison metrics: Precision, Recall, and balanced F-Score. In the case of overlapping or partial matches, we penalize all tools by adding $\frac{1}{2}FP$ (False Positive) and $\frac{1}{2}FN$ (False Negative) to the precision and recall equations (e.g., if the tool spots "The Louisiana" instead of "Louisiana").

As we aim to extract location names that are inside the area of interest, we evaluate all tools based on the category of the extracted location in our annotation scheme. For the inLOC mentions, we count all hits and misses of a tool and ignore all hits when the category of the extracted location is outLOC or ambLOC. However, we take a particularly conservative approach and additionally penalize LNEx for extracting location names of outLOC and ambLOC categories, counting them as false positives (FPs) as our tool is not supposed to extract these.

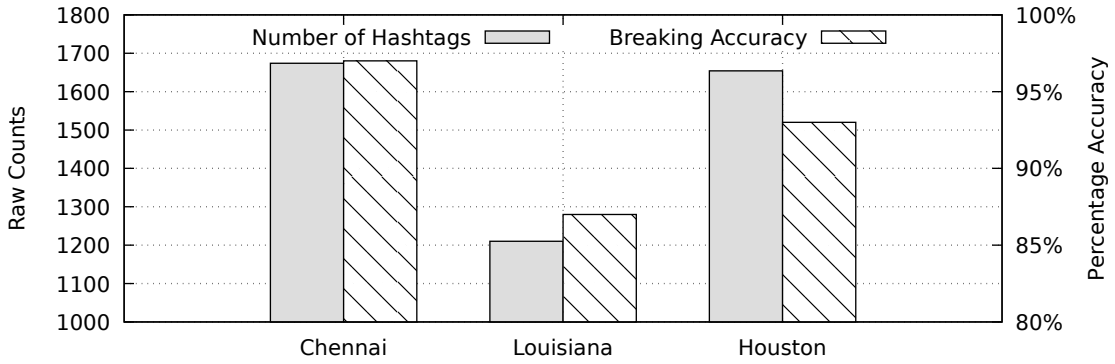


Figure 2.4: Hashtag breaking accuracy.

2.4.5 Spell Checking

The literature on location names emphasizes the effect of misspellings in tweets [40, 75, 125]. SymSpell fixed the misspellings of 105 tokens from Chennai (3.64%), six from Louisiana (0.35%), and 21 from Houston (1.22%). While this led to a 1% increase in recall, the F-Score decreased by 2% on average due to the influence of increased false positives on precision. In the final system, we opted to exclude the spelling corrector component.

2.4.6 Hashtag Breaking

Using the annotated data, we evaluated the performance of the hashtag breaking component only on the hashtags that contain locations (see Figure 2.4). The accuracies were 97%, 87%, and 93% for Chennai, Louisiana, and Houston respectively, reduced due to examples such as, “#lawx” which was broken into “law” and “x”, instead of “la” and “wx”, because the combined probability of the word “law” and the character “x” is higher than the other combinations.

2.4.7 Picking a Gazetteer

The augmentation and filtering of gazetteers improved the F-Scores (See Figure 2.5). After this process, combinations of gazetteer sources had similar performance (based on the average F-Score) where the difference between the worst and the best was around 0.02 F-Score units (see Figure 2.6). In the final system, we relied on OSM, which performed the best. Moreover, DBpedia is not focused on geographical information; therefore, it does not contain the metadata useful for the system's future use (e.g., extents and full addresses). Also, OSM has more fine-grained locations and more accurate geo-coordinates than Geonames [39].

2.4.8 Comparing LNEx with Other Tools

We compared LNEx with the following ten tools, either using their public APIs or using their available open-source implementations. The tools mentioned in the related work section and not mentioned below are not available online for our evaluation and comparison. However, many of them retrained and used Stanford NER or OpenNLP. Therefore, we also retrained both for comparison.

1. **Commercial Grade**: Google NL[10], OpenCalais[11], and Yahoo! BOSS PlaceFinder[12]. All of these tools have REST APIs and are black box tools that use Machine Learning techniques. However, Google mentions that their API is based on deep neural network technology.

2. **General Purpose NER**: Stanford NER (SNER) [13] and OpenNLP Name Finder[14]. To spot named entities in texts, SNER learns a linear chain Conditional Random Field (CRF) sequence model [35], while OpenNLP uses the maximum entropy (ME)

[10] https://cloud.google.com/natural-language/
[11] http://www.opencalais.com/
[12] https://developer.yahoo.com/boss/geo/
[13] https://nlp.stanford.edu/software/CRF-NER.html
[14] https://opennlp.apache.org/

framework [10]. We trained both tools interchangeably on our annotated datasets in addition to all the data from W-NUT 2016[15]. As the W-NUT dataset contains other entity classes besides locations, we retained the annotated locations and unified the other classes we recognize as locations (i.e., geo-loc, company, facility) into one type and ignored the rest. We used the LNEx-OSM gazetteer's features while training SNER. Additionally, we used DBpedia Spotlight [85].

3. **Twitter NLP**: OSU Twitter NLP [109][16] and TwitIE-Gate [12][17]. Both are pipelined systems with POS-tagging followed by NER. They adapted CRF-based POS taggers and trained them on manually annotated tweets as part of the full pipeline. TwitIE-GATE also supports normalization, gazetteer lookup, and regular expression-based tagging. For a fair comparison, we also augmented them with LNEx OSM gazetteers.

4. **Twitter Location Extraction**: Geolocator 3.0 [40][18] and Geoparsepy [86][19]. Geolocator 3.0 uses a tweet-trained CRF classifier and other rule-based models to extract street names, building names, business names, and unnamed locations (i.e., location names containing a category such as "School"). Also, it only geocodes the extracted toponyms and business names. Geoparsepy is the implementation of the work in [86].

All tools have been evaluated using the same metrics and on the same annotated data. In the case of hashtags, we count all hits for all tools, and when a tool misses, we penalize only the ones that were designed to break hashtags (namely, TwitIE-Gate[20] and LNEx). Additionally, we consider all spotted mentions from PlaceFinder, Geolocator 3.0, Geop-

[15] http://noisy-text.github.io/2016
[16] https://github.com/aritter/twitter_nlp
[17] https://gate.ac.uk/wiki/twitie.html
[18] https://github.com/geoparser/geolocator-3.0
[19] https://pypi.python.org/pypi/geoparsepy
[20] http://rebrand.ly/gatea336f

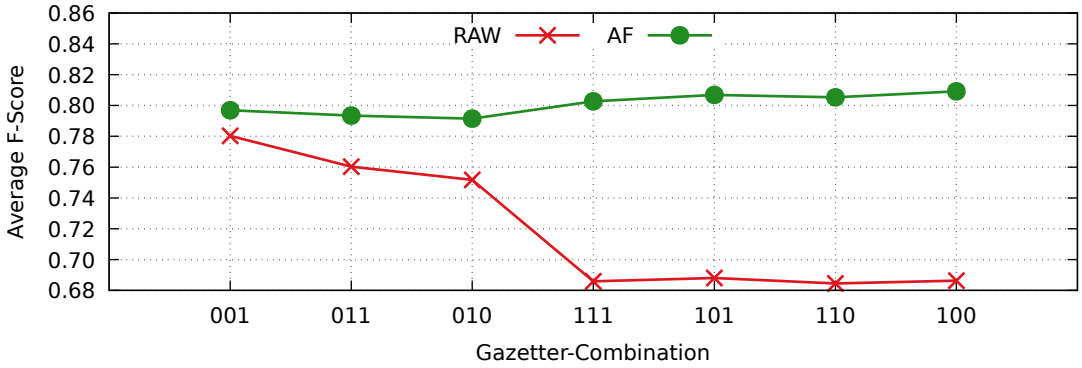


Figure 2.5: Raw vs. augmented and filtered (AF) gazetteers combinations. Each of the seven combinations is a subset of {OSM Geonames DBpedia}.

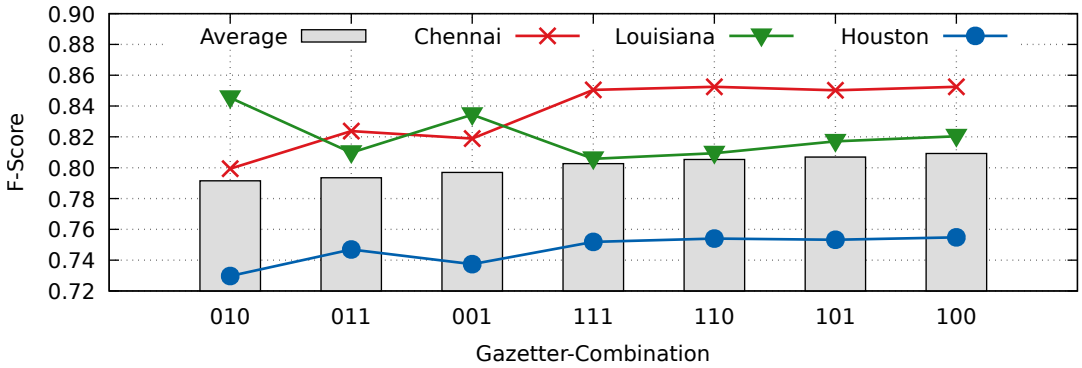


Figure 2.6: Gazetteer combinations performance. Each of the seven combinations is a subset of {OSM Geonames DBpedia}. E.g., 100 stands for {OSM} while 011 stands for {Geonames DBpedia}.

Table 2.5: Types considered as location classes per tool.

Tool Name	**Entity Types of Location Names**
Google NLP	Location, Organization
OpenCalais	City, Company, Continent, Country, Facility, Organization, ProvinceOrState, Region, TVStation
DBpedia Spotlight	Place, Organization
OSU TwitterNLP	Geo-Location, Company, Facility
TwitIE-Gate	Location, Organization

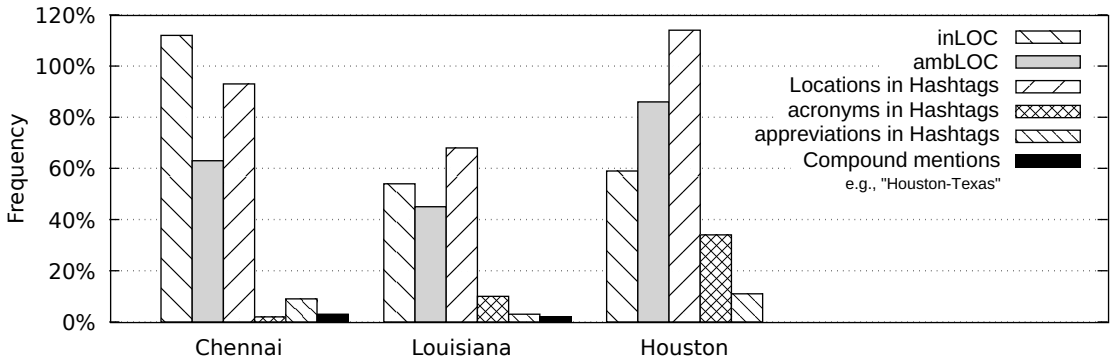


Figure 2.7: Random sample evaluation showing the prevalence of the challenges and the frequency of the different form types discussed in Section 2.2.

arsepy, SNER, and OpenNLP as location names. But for other tools, we consider only the entity types in Table 2.5 as location entities. As we trained SNER and OpenNLP on a single class (i.e., location names), the two tools extract entities of this type only.

Figure 2.7 shows that the three corpora exhibit different challenges to location extraction (see the full list of challenges in Section 2.2). Nevertheless, LNEx outperformed all other tools on all datasets in terms of F-Score, and the average F-Score (see Table 2.6). LNEx also achieved statistical significance across the board when using the Mann-Whitney U test, with a $P < 0.05$. Furthermore, LNEx showed stability on the test and development sets from Louisiana with only a 0.2% F-Score reduction and around a 2.6% reduction on the test set from Chennai.

The augmentation and filtering method significantly improved the average F-Score from 0.69 to 0.81. However, limitations of the gazetteer augmentation and filtering methods did contribute to lowering precision. For example, on average, around 5% of the extracted location names were outLOC and ambLOC, mistakenly extracted from Chennai, Louisiana, and Houston tweets. Example errors include the augmentation of location names such as "The x Apartments" to "The Apartments", causing LNEx to extract the phrase "The Apartments" as an actual full location name. Fixing such limitations should contribute to around a 2% F-Score improvement on average. The inability of the similar OSM-gazetteer-

Table 2.6: Location extraction tools vs. LNEx with a raw (RawGaz), and augmented and filtered gazetteer (AFGaz). LNEx improves its performance by using AFGaz and outperforms all tools with at least 33% F1-Score improvement.

	Datasets									
	Chennai			Louisiana			Houston			AVG
	P	R	F	P	R	F	P	R	F	F
Google NLP	0.40	0.49	0.44	0.55	0.75	0.64	0.39	0.51	0.44	0.51
OpenCalais	0.43	0.10	0.17	0.81	0.77	0.78	0.62	0.35	0.45	0.47
DBpedia Spotlight	0.31	0.44	0.36	0.57	**0.88**	0.70	0.35	0.53	0.42	0.50
Yahoo! PlaceFinder	0.67	0.39	0.49	0.83	0.80	0.81	0.64	0.42	0.50	0.61
Stanford NER	0.72	0.29	0.41	0.78	0.42	0.55	0.74	0.32	0.45	0.47
OpenNLP	0.55	0.15	0.24	0.62	0.19	0.29	0.60	0.23	0.34	0.29
OSU TwitterNLP	0.74	0.40	0.52	**0.84**	0.69	0.76	0.66	0.39	0.49	0.59
TwitIE-Gate	0.51	0.36	0.43	0.66	0.84	0.74	0.35	0.39	0.37	0.52
Geolocator 3.0	0.43	0.54	0.48	0.32	0.71	0.44	0.38	0.58	0.46	0.46
Geoparsepy	0.41	0.28	0.33	0.45	0.72	0.55	0.44	0.46	0.45	0.45
LNEx-RawGaz	0.80	0.78	0.79	0.51	0.80	0.62	0.63	0.66	0.64	0.69
LNEx-AFGaz	**0.91**	**0.80**	**0.85**	0.83	0.81	**0.82**	**0.87**	**0.67**	**0.76**	**0.81**

based approach by [86] to filter and augment the gazetteers hindered the performance of Geoparsepy as the tool was unable to extract location mentions using the raw gazetteers or mistakenly extract false positive locations.

We trained Stanford NER (SNER) and OpenNLP to emulate their use in other studies mentioned in Section 2.5. Performance was calculated by interchangeably training them using three datasets at a time and testing on the fourth one (the gazetteer of the area of the test data was also used in training the SNER models). We always used the W-NUT '16 dataset to train the models with more than 10k tweets each time.

We observed that the ill-formatted text of tweets with ungrammatical text and missing orthographic features impact the F-Score of tools we compared with LNEx. While the performance of each tool differs, we observed that Google heavily relies on orthographic features and expects grammatical texts (although it scored a 0.38 average F-Score). Additionally, TwitIE-GATE was not always successful in extracting location names from hashtags or text even if they are part of the gazetteers that we added to the tool. Finally, OpenCalais extracts only well-known location names of coarser granularity than street and building levels unless a location has an attached location category (e.g., school or street).

2.4.9 Illustrative Examples

Table 2.7 shows the comparative handling of three tweets, one each from Chennai, Louisiana, and Houston datasets, covering most challenges faced by all tools. The location name "Oxford school" allowed us to examine if a tool relies on capitalization for delimitation. Only OpenCalais, Geolocator and LNEx were able to extract the name correctly while the rest either partially extracted it or missed it. For example, PlaceFinder extracted "Oxford" and geocoded it with the geocodes of "Oxford city" in England. Although we trained SNER and OpenNLP on the same datasets, OpenNLP extracted "Oxford" while SNER did not, which suggests that the cue word "near" was insufficient evidence for SNER to spot at least

Table 2.7: Example tool outputs: the bracketed bold text is the identified location names, and braces highlight the types from Section 2.2.

Original Text's Manual Annotations & Types	T12 T16 **(sou th kr koil street)** near **(Oxford school)** . **(west mambalam)** T13 T15 T13
	T7 T2 T8 We r lucky where I am in **(New Iberia)**. #PrayFor **(Louisiana)** #(**la**)wx
	Didn't **(Houston)** have a bad flood last year now again poor **(htown)** T7 T11
Google NLP	sou th kr (**koil street**) near (**Oxford**) school.west (**mambalam**)..
	We r lucky where I am in (**New Iberia**). #PrayForLouisiana #lawx
	Didn't (**Houston**) have a bad flood last year now again poor htown
OpenCalais	sou th kr koil street near (**Oxford school**).west mambalam..
	We r lucky where I am in New Iberia. #PrayForLouisiana #lawx
	Didn't (**Houston**) have a bad flood last year now again poor htown
DBpedia Spotlight	sou th kr koil street near (**Oxford**) school.west (**mambalam**)..
	We r lucky where I am in (**New Iberia**). #PrayForLouisiana #lawx
	Didn't (**Houston**) have a bad flood last year now again poor htown
Yahoo! PlaceFinder	sou (**th**) kr koil street near (**Oxford**) school.west mambalam..
	We r lucky where I am in (**New Iberia**). #PrayForLouisiana #lawx
	Didn't Houston have a bad flood last year now again poor htown
Stanford NER	sou th kr koil street near Oxford school.west mambalam..
	We r lucky where I am in (**New Iberia**). #PrayForLouisiana #lawx
	Didn't Houston have a bad flood last year now again poor htown
OpenNLP	sou th kr (**koil street**) near (**Oxford**) school.west mambalam..
	We r lucky where I am in (**New Iberia.**) #PrayForLouisiana #lawx
	Didn't Houston have a bad flood last year now again poor htown
OSU TwitterNLP	sou th kr koil street near (**Oxford**) school.west mambalam..
	We r lucky where I am in (**New Iberia**). #PrayForLouisiana #lawx
	Didn't Houston have a bad flood last year now again poor htown
TwitIE-Gate	sou th kr koil (**street**) near (**Oxford**) school.(**west mambalam**)..
	We r lucky where I am in New Iberia. #PrayForLouisiana #lawx
	Didn't Houston have a bad flood last year now again poor htown
Geolocator 3.0	(**sou th**) (**kr**) (**koil**) street near (**Oxford school**).(**west mambalam**)..
	We r lucky where I am in (**New Iberia**). #PrayForLouisiana #lawx
	Didn't (**Houston**) have a bad flood last year now again poor (**htown**)
Geoparsepy	sou (**th**) (**kr**) koil street near (**Oxford**) school.west mambalam..
	We r lucky where I am in (**New (Iberia)**). #PrayForLouisiana #lawx
	Didn't (**Houston**) have a bad flood last year now again poor htown
LNEx	sou th kr koil street near (**Oxford school**).(**west mambalam**)..
	We r lucky where I am in (**New Iberia**). #PrayFor(**Louisiana**) #lawx
	Didn't (**Houston**) have a bad flood last year now again poor htown

"Oxford". Correspondingly, as "New Iberia" is a correctly capitalized full location name, almost all tools were able to extract it. However, TwitIE-Gate missed it although it is part of the gazetteer we added to the tool, and Geoparsepy extracted Iberia in addition to the full mention, not favoring the longest mention as LNEx. OpenCalais is a black box, so we do not know why it failed.

Regarding T3-T5 annotations, LNEx and TwitIE-Gate are designed to break hashtags, but TwitIE-Gate was not able to extract any locations from the hashtags in the table. LNEx extracted "Louisiana" but was not able to extract "la" from "#lawx" due to the statistical method that broke the hashtag into "law" and "x" as this combination is more probable. Only Geolocator was able to extract the Houston nickname "htown". In the future, a dictionary of region-specific acronyms, abbreviations, and nicknames can augment LNEx's region-specific gazetteers.

Google NLP does not handle T6. However, adding space between the dot and "west" to create ". . . school. west . . . ", results in the extraction of "west mambalam" but omits "Oxford school". Google NLP relies on capitalization and so that changing the case of "s" to create "Oxford School" does help. OpenCalais cannot extract "west mambalam" despite fixing all grammatical mistakes, normalizing the orthographic features, and even introducing cue words. The tool only extracts well-known location names of coarser granularity than street and building levels unless they have an attached location category (e.g., school or street). PlaceFinder, on the other hand, tries to find geocodable location names in text. Therefore, the tool extracts "th" as the country code of Thailand and "Oxford" as the city in England. Hence, geocoding is influencing some of the mistakes of the tool.

The correct location for "sou th kr koil street" is "South K R Koil Street". This location mention has two problems: a misspelling ("sou th") and a T6-compound mention problem ("kr"). LNEx system cannot rejoin a word partitioned by misspelling, or break words other than hashtags. LNEx examines each token individually and joins them only if they are valid n-grams in the language model (see Section 2.3.1). Moreover, this particular

record is missing from our gazetteers. Without gazetteer repair, LNEx would not extract it. Moreover, this particular record is missing from the OSM gazetteer, exacerbating the problem.

2.5 Related Work

The majority of the location-related techniques in the literature are focused on locating social media users and then inferring the location of the authored texts. These studies rely heavily on tweet metadata, such as the profile location field and the attached geocoordinates. Also, some studies analyze a user friend's network to infer the location of the user [113, 31, 106]. In contrast, our approach seeks to extract location mentions from the text to localize referents and support location-aware applications.

Extracting location names from texts is a special kind of Named Entity Recognition (NER) where the extracted entities are only of type location. Here, we extract entities of type location only. When describing other general-purpose entity extractors (i.e., the tools that extract entities of other types besides locations like Google NL, TwitIE-GATE), we focus only on the entity types that represent a location (e.g., Organization, Facility).

Twitter messages (tweets) lack features exploited by mainstream NLP tools. Informality, ill-formed words, irregular syntax, and non-standard orthographic features of tweets challenge such tools [55]. We agree with [8] that some issues might be exaggerated. Indeed we found that spelling corrections only contributed to 1% recall improvement. Therefore, text normalization alone is insufficient for NER [26]. Specially designed tools such as [109, 40] use pipelined systems of POS tagging followed by NER. The latter also perform Regex tagging, normalization, and gazetteer lookup to deal with these challenges.

Relying on the orthographic features for POS tagging or Regex tagging, previous methods extract locations from the text chunks and phrases of sentences using the following techniques:

1. **Gazetteer search or n-gram matching**: [65] and [40] use a gazetteer matching technique that relies on a segment-based inverted index. [125] use an exhaustive n-gram technique. [86] use location-specific gazetteers for matching phrases from tweets. TwitIE-GATE uses a gazetteer lookup component. None of these techniques deal with the important issue of gazetteers' auxiliary content and noise.

2. **Handcrafted rules**: [132] and [75] use pattern and Regex matching that rely on cue words or orthographic features for POS-tagging. TwitIE-GATE adapts rules from ANNIE [22] for extraction.

3. **Supervised Methods**: *Tweet-trained models*: The majority of the methods trained SNER on tweets [40, 138] or other CRF implementations [71, 57, 48, 63] or retrained OpenNLP [68]. *News-trained models*: [75] use tools like SNER and OpenNLP.

4. **Semi-supervised methods**: [53] use beam search and structured perceptron for extraction and linking to Foursquare entities. However, they did not address the noise that is prevalent in such sources (e.g., "my sofa" or "our house") [24].

Handcrafted rule-based methods are hard to develop, and supervised methods are labor-intensive. As our specific application is time-critical, annotating tweets during an ongoing disaster is likely infeasible, and is undoubtedly ineffective for real-time processing. While many state-of-the-art methods retrained different location extraction models, I showed that these models need a considerable amount of data, and with around $13k$ tweets, the best model still scored on average 0.47 F-Score.

The closest work to ours is TwiNER [64] and LEX [30]. Both use Microsoft Web n-grams (which capture language statistics) for chunking, but the former uses DBpedia for entity linking. For example, [64] used Microsoft web N-grams to segment tweets and then match the valid segments with DBpedia pages as a way to recognize segments as entities. On the other hand, LEX works with web data and relies heavily on capitalization, which is

not a reliable feature in tweets. In contrast, our method uses a region-specific gazetteer for delimitation and linking.

Finally, a few other methods extract locations from hashtags. [75] uses a statistical hashtag breaker technique similar to ours. [53] removes only the # symbol and treats the hashtag as a unigram. [138] uses a greedy maximal matching method for breaking. TwitIE-GATE uses two methods for hashtag breaking: a dynamic programming-based method for finding subsequences and a camel-case-based method for tokenization.

2.6 Conclusions, Limitations, and Next Steps

Does an augmented region-specific gazetteer successfully extract location names from a targeted text stream? I demonstrated that the answer is a resounding yes. LNEx accurately spots locations in text, relying solely on statistical language models synthesized from augmented and filtered region-specific gazetteers. It outperforms state-of-the-art techniques and mainstream location name extractors. By exploiting the knowledge in a gazetteer, we retain the benefits of n-gram matching to access location metadata. LNEx does not employ any training and does not depend on syntactic analysis or orthographic conventions. We compensate for limitations in fixed phrase matching with gazetteer augmentation and filtering. Although we do not solve the disambiguation problem here, still the geo/-geo ambiguity is reduced by preserving the spatial (and cultural) context through location-specific gazetteers. Furthermore, systematic gazetteer augmentation ties legitimate variants to known locations, minimizing potential ambiguity.

Although our statistical method outperforms other state-of-the-art approaches, it still can benefit from textual cues that help in detecting ambiguous location names such as proper names instead of always filtering them out using stop word lists. Additionally, as the linking procedure drives the method, it does not extract location names missing from gazetteers (e.g., “our house”). It presents an effective precision-recall trade-off apparent in

the F-Score. In Chapter 4, I demonstrate the use of a dictionary-based technique based on the same n-gram model as in this chapter. However, in Chapter 4, I show how effective the use of knowledge and a human-in-the-loop can be in partially overcoming the difficulties faced by the context-agnostic dictionary-based NER approach that uses incomplete and noisy dictionaries.

Sparse Entity Extraction

Extracting sparse named entities from textual data using supervised techniques faces the following challenges: (1) sparsity of labels and the high cost of data labeling (requiring massive amount of data to learn accurate models), (2) quality of annotations (posing a limitation on who can perform the annotation task and other related issues such as developing, expensive but clear, guidelines), and (3) selection of an appropriate evaluation criteria (mainly to avoid the additional cost of having a held-out set for evaluation). In this chapter, I present our SpExtor framework [4][1], which integrates entity set expansion (ESE) and active learning (AL), to reduce the annotation cost of sparse data and provide an online evaluation method as feedback. This incremental and interactive learning framework allows for rapid annotation while maintaining high accuracy. Consistent with the theme of this dissertation, this method also lowers the cost of building a robust entity extraction mechanism exploiting the knowledge of a subject matter expert (SME) and other intelligent sampling techniques (i.e., ESE and AL).

I also present the evaluation of the framework on three publicly available datasets and show that it drastically reduces the cost of sparse entity annotation by an average of 45% to 85% while reaching 1.0 and 0.9 F-Score (on the same sentences pool), respectively. Moreover, the method exhibits robust performance across all datasets.

[1] Part of this work was also patented in the US. [32].

3.1 Introduction

Supervised machine learning methods exploit the inductive power of NER models learned from gold annotations labeled for all mentions of an entity class/type in unstructured text (e.g., Location Names, Proteins, or Auto Parts IDs). These techniques have proven to be effective in extracting named entities [135]. However, the prohibitive cost of obtaining a large volume of labels for training makes them unattractive and hard to use in realistic settings where resources may be scarce or costly to acquire (e.g., the time cost that aviation engineers need to annotate engine maintenance data). In this chapter, I present our SpExtor framework that addresses the dire need for a practical solution that exploits the strengths of the supervised sequence labeling techniques [137] while reducing the high cost of annotation.

Building solid entity extraction models that are based on, for example, conditional random field [35] or other sequence labeling techniques require two things: rich feature sets (for capturing the semantics of the entity class of interest) and a large number of (accurately labeled) training samples. To featurize sentences, we use the comprehensive list of features provided by CoreNLP's NERFeatureFactory [81]. As to labeling, we use a smart sampling technique using ESE (with graph embedding) and AL to decrease the number of labeled sentences required for training models without sacrificing the accuracy of the extraction.

Our approach is similar to the work proposed by [128] in that we aim to annotate all mentions of a single entity class in corpora while relaxing the requirement of full coverage. Our design choice lowers annotation cost by using a realistic approach to flexible stopping criteria. It annotates a corpus without requiring annotators to scan all sentences. However, it differs from [128] in that we use ESE to learn semantics and retrieve similar entities by analogy to accelerate learning (therefore using ESE as a smart sampling technique). Moreover, our approach provides Fast (FA), Hyper-Fast (HFA), and Ultra-Fast (UFA) auto-annotation modes for rapid annotation (The full code is available online at `https://github.com/halolimat/SpExtor`).

In the upcoming sections, I describe our incremental learning solution developed using AL, and how we use ESE, which is based on graph embedding to accelerate the annotation. Finally, I present the evaluation of the approach on publicly available datasets demonstrating its effectiveness.

3.2 Incremental and Interactive Learning

In a realistic setting, domain experts have datasets from which they want to extract entities, and then use them to build various applications, starting from inverted indexes or catalogs, to targeted analysis, e.g., to obtain sentiments. Next, I describe how we used AL to accelerate learning, auto-annotate sentences, and to provide feedback for flexible stopping criteria.

3.2.1 Active Learning (AL)

Supervised sequence labeling techniques require labeled data to induce a model that can be used to extract entities from unstructured text. Traditional supervised machine learning frameworks randomly or sequentially query SMEs to label sentences from a pool of unlabeled sentences $\mathcal{U}$. The labeled instances are then placed in the labeled pool $\mathcal{L}$ once labeled. These frameworks usually keep querying users for labels and adding the sentences to $\mathcal{L}$ until $\mathcal{U}$ is empty. Using this labeling method is quite expensive, requiring us to label the whole sentence pool without any chance of minimizing the wasteful effort that might result from duplication in the semantics of annotations without improving the model's accuracy.

Active learning minimizes wasteful effort by intelligently sampling sentences from $\mathcal{U}$ based on some selection criterion and then querying the user to label the sentences, learn a model from $\mathcal{L}$, and determine the next batch of sentences for labeling. Selection criteria

are usually implemented with the help of the learned model, which factors the knowledge learned from $\mathcal{L}$ in the previous step to choose the sentences for labeling in the current step. For example, the top b sentences can be sampled from a ranked list of sentences from $\mathcal{U}$ based on the model's accuracy on tagging them.

We adopt the linear-chain Conditional Random Field (CRF) implementation by [35] to learn a sequence labeling model to extract entities. However, we use pool-based AL for sequence labeling [115] to replace the sequential or random samplers during online corpus annotation. Our iterative AL-enabled framework requires a pre-trained/base model to sample the next batch of sentences to be labeled. As stated before, this iterative sampling allows us to reach higher accuracies with fewer data points by incrementally factoring the new knowledge encoded in the trained models. This more informative sampling is achieved due to the added requirement of model training on all previously annotated sentences as well as new batches of annotated sentences[2]. In Section 3.3, I show how learning the base model by annotating sentences sampled using the entity set expansion (ESE) method that we developed is better than learning a model from a random sample.

As for the selection criterion, which is the method needed for selecting the candidate sentences to be labeled by annotators based on measuring their informativeness, we employ an uncertainty-based sampling technique, the n-best sequence entropy method [60]. While the default inductive behavior of a pre-trained CRF model is to provide one sequence (the most probable one), we instead employ an n-best sequence method which uses the Viterbi algorithm to return, for each sentence, the top n sequences with their probabilities. We then query the annotator to annotate b number of sentences (equal to a pre-selected batch size) with the highest entropies [115] calculated using the following equation:

[2]We train all CRF models using the default set of features for CoNLL 4 class - https://rebrand.ly/corenlpProp

$$nSE(m, s) = -\sum_{\hat{y} \in \hat{Y}} p_m(\hat{y}|s;\theta)\ log\ p_m(\hat{y}|s;\theta) \tag{3.1}$$

where m is the trained CRF model, $s \subset \mathcal{U}$, $\hat{Y}$ is the set of n sequences, and θ is the features' weights learned by the model m during the supervised training.

3.2.2 Annotation Modes

In this framework, we devise four AL-enabled annotation modes:

1. **EAL**: Entity set expansion for learning the base model (see Section 3.3) plus active learning for the remaining steps. For sampling, this mode uses model confidence on sentences estimated using nSE.

2. **FA**: Fast auto-annotation mode plus EAL.

3. **HFA**: Hyper Fast auto-annotation mode plus EAL.

4. **UFA**: Ultra Fast auto-annotation mode plus EAL.

The last three annotation modes use a thresholded auto-annotation method that we developed to accelerate the annotation procedure. Auto-annotation employs a pre-trained CRF model m to annotate all unlabeled sentences in the pool. Then, we accept the annotation of sentences from the model if they satisfy the following condition: **if** $\frac{SE_1(s)}{SE_2(s)} \leq t$, where $SE_i(s)$ is the entropy of the sequence i of the sentence s and t is a predefined threshold representing the desired marginal ratio between the entropies of the two sequences 1 and 2. We chose t empirically. We found that the thresholds below 0.10 were tight (resulting in no auto-annotations), and the ones above 0.20 were large (resulting in many incorrect

annotations). Therefore, we chose 0.10, 0.15, and 0.20 for the FA, HFA, and UFA auto-annotation modes, respectively. In Section 3.4.4, we evaluate the effectiveness of these annotation modes.

3.2.3 Interactive Learning

Active learning is useful in sampling the next batch of sentences so that labeling them would increase model accuracy. However, AL alone will not be useful without a stopping criterion needed to stop annotating more sentences from the pool $\mathcal{U}$. Therefore we designed an online evaluation method σ (Equation 3.3) that provides feedback to annotators on the confidence of a model m for a given sentences pool $\mathcal{S}$, where $\mathcal{S} = \mathcal{U} \cup \mathcal{L}$. This feedback is an alternative to the F1-measure and is very valuable in the absence of a gold standard dataset that could otherwise provide this kind of feedback.

$$\overline{nSE}(m, \mathcal{S}) = \frac{1}{|S|} \sum_{s \in \mathcal{S}} nSE(m, s) \tag{3.2}$$

$$\sigma = 1 - \overline{nSE}(m, \mathcal{S}) \tag{3.3}$$

As nSE is based on uncertainty-based measurement (entropy) [119], the mean of all nSEs (i.e., $\overline{nSE}(m, \mathcal{S})$) would, therefore, provide us with the uncertainty of the model m on the labeling of the sequences of words in the sentences from the pool $\mathcal{S}$. As σ is equal to 1 minus the uncertainty of the model, then σ would contain the certainty of the model, which gives the annotator a clearer picture of whether to keep the model as is or learn a new model, interactively. We use σ both during the incremental learning steps from the same sentence pool and to test the model's accuracy/confidence on unseen sentences. Therefore, using this feedback, we can decide to stop annotating from a certain sentence pool and augment it with sentences from a new pool, or simply stop annotating and train a final

model. Consequently, deciding to annotate new sentences that contain novel entities helps reduce the effect of data sparsity and increases the accuracy of models, which highlights the importance of this online feedback method.

$$Improvement(m_1, m_2, \mathcal{S}) = 1 - \frac{\overline{nSE}(m_2, \mathcal{S})}{\overline{nSE}(m_1, \mathcal{S})} \tag{3.4}$$

Equation 3.4 helps in knowing the percentage improvement a new model m_2 exhibits in comparison with the old model m_1 when trained on more sentences from $\mathcal{U}$. While Equation 3.3 can alone be used to help the extraction from a new unseen dataset, this equation can be used to show the improvement of the accuracy of the model with new batches of labeled sentences from the same sentence pool. Hence, given a small improvement or learning rate, the annotator can stop the annotation or may decide to use another pool of sentences and see if a significant improvement can be achieved.

We compare our online evaluation method with the estimated coverage method in [128], which computes the expected number of entities in $\mathcal{S}$ as follows:

$$EC(\mathcal{S}) = \frac{E_{\mathcal{L}}}{E_{\mathcal{L}} + \sum_{u \in \mathcal{U}} E_u} \tag{3.5}$$

where $E_{\mathcal{L}}$ is the number of annotated entities in the labeled sentences $\mathcal{L}$, and E_u is the expected number of entities in the unlabeled sentence u. E_u is calculated by summing the probability of each entity in all of the n-best sequences.

3.3 Entity Set Expansion for Base Model Learning

Active learning requires a base (pre-trained) model to start sampling sentences for annotation [115]. Learning the base model from annotated sentences sampled using sequential or random sampling can be very expensive due to a problem we call *labeling starvation*, i.e., querying annotators to label sentences containing a low frequency of entities from the

desired class. As our framework imposes sentence-level and single-entity-class annotation requirements (to improve tractability and reduce cost), that causes the labeling starvation problem. For example, in Table 3.1, only 8% of the sentences have virus entities. Therefore, we developed the Entity Set Expansion (ESE) method that incorporates and exploits the semantics of the desired entity class to sample sentences more informatively that are likely to have entities of the desired class.

We assume that all noun phrases are candidate entities and extract all of them from the unstructured text.[3] Then, we record five features for each noun phrase (np) and model the data as a bipartite graph with noun phrases on one side and features on the opposite side, as described below.

3.3.1 Noun Phrase Extraction (NPEx)

The first step of our method is to extract noun phrases, which we consider as candidate entities. We POS tag and parse sentences to the tagged sequence form: $[(w_i\ pos_i)\ \forall w \in W]$ (e.g., "`Kankkunen NNP was VBD runner-up JJ in IN his PRP$ Toyota NNP . .`"). Then, we use the following regular expressions to extract the noun phrases:

```
NPs = JJs + NNs + CDs
JJs = (?:(?:[A-Z]\\w+ JJ )*)
NNs = (?:[^\\s]* (?:N[A-Z]*)\\s*)+
CDs = (?:\\w+ CD)?
```

where NPs are noun phrases, JJs are adjectives, NNs are nouns, and CDs are cardinal numbers. The final set of noun phrases from the above example contains {"Kankkunen", "Toyota"}.

[3] While not all candidate noun phrases are positive examples of an entity, a human-in-the-loop would interactively give feedback to the system by choosing the positive ones.

3.3.2 Featurization

To draw relationships between the extracted noun phrases to retrieve them by analogy, we automatically extract and record five kinds of features for each noun phrase that we can extract from the text (see Section 3.3.1). The list of features follows:

① **Lexical Features (LF):** We extract these from the surface form of words and sequences of letters in text. The following are the two types of lexical features we extract:

- **Orthographic Form (OF)**: We abstract OF features from the actual word and obtain its type (i.e., numeric, alpha, alphanumeric, or other). Additionally, we classify the word as *all upper*, *all lower*, *title case*, or *mixed case*.
- **Word Shape (WS)**: We define WS to abstract the patterns of letters in a word as a short/long word shape (SWS/LWS) features. In LWS, we map each letter to "L", each digit to "D", and retain the others. On the other hand, for SWS, we remove consecutive character types. For example, LWS(ABC-123) $\rightarrow$ "LLL-DDD" and SWS(ABC-123) $\rightarrow$ "L-D".

② **Lexico-Syntactic Features (LS):** We use a skip-gram method to record the explicit LS-patterns surrounding each NP, i.e., for each np in s we record the pattern $w_{i-1} + np + w_{i+1}$, where $w_{(.)}$ are the two words that precede and follow np.

③ **Syntactic Features (SF):** We use dependency patterns to abstract away from the word-order information that we can capture using contextual and lexico-syntactic features. We use Stanford's English_UD neural network-based dependency parser [17] to extract universal dependencies of noun phrases. For each np in s, we record two dependency patterns of the noun phrase that serve in *governor* and *dependent* roles.

④ **Semantic Features (SeF):** To capture lexical semantics, we use WordNet [88] to get word senses and draw sense relations between noun phrases if they have the same sense class.

⑤ **Contextual Features (CF):** To capture latent features, we use a Word2Vec embedding model [87] trained on the sentence pool S as the only bottom-up distributional semantics method. We exploit the word-context co-occurrence patterns learned by the model to induce the relational similarities between noun phrases.

The use of semantic (i.e., word senses) and contextual (i.e., word embeddings) features on sparse or domain-specific data (such as enterprise data) is not enough and sometimes impossible due to unavailability [126]. Therefore, we tried to use diverse features in a complementary manner to capture as many meaningful relations between potential entities as possible. For example, the absence of SeF for domain-specific entities, such as protein molecules or parts numbers, is compensated by the use of WS-LF (e.g., by drawing a relation between the noun phrases "IL-2" and "AP-1").

3.3.3 Feature-Graph Embedding

Here, I discuss the construction of the bipartite graph, which we use to allow for calculating the similarity between noun phrases by modeling the edges from multi-modal edge weights. We model edges in the bipartite graph by assigning a weight w between each pair of a noun phrase n and a feature f using one of the following:

$$w^1_{n,f} = C_{n,f} \tag{3.6}$$

$$w^2_{n,f} = log(1 + C_{n,f})[log|N| - log(|N|_f)] \tag{3.7}$$

$$w^3_{n,f} = log(1 + C_{n,f})[log|N| - log(\sum_{\hat{n}} C_{\hat{n},f})] \tag{3.8}$$

where $C_{n,f}$ is the co-occurrence count between n and f, $|N|$ is the number of noun phrases in our dataset, $|N|_f$ is the size of the set of noun phrases with a feature f, and $\sum_{\hat{n}} C_{\hat{n},f}$ is the sum of all noun phrase co-occurrences with the feature f. Equations 3.7 and 3.8 are two variations of the term frequency–inverse document frequency (TFIDF) measurement, intended to reflect the importance of a feature in the embedding graph. The latter is adapted to weigh the edges in [120, 110]. Rong et al. [110] found Equation 3.8 to work better than PMI. Therefore, we did not implement and compare it with the other weighing methods here.

Algorithm 2: The entity set expansion algorithm we developed to return a ranked list of candidate entities to bootstrap the learning of the base model needed for active learning.

Data: e: input seed; S: text sentences

Result: $N = \{n\}$: ranked similar noun phrases

Start

$\hat{N}$ **=** $\emptyset$**;** `// all noun phrases`

F **=** $\emptyset$**;** `// all features`

for $s \leftarrow S$ **do**

  $\hat{N} = \hat{N} \cup$ **ExtractNounPhrases(**s**);**

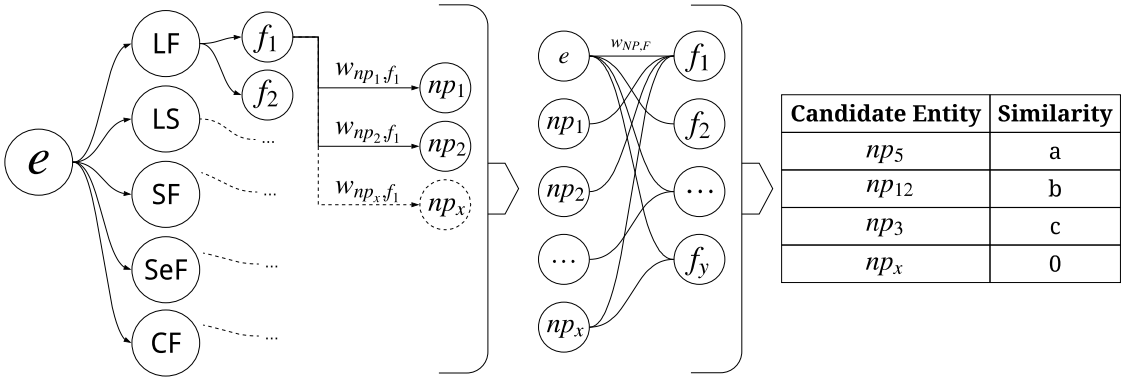
  $F = F \cup$ **Featurize(**$\hat{N}$**);** `// Section 3.3.2`

end

$G =$ **BuildBipartiteGraph(**$\hat{N}$**,** F**);** `// Section 3.3.3`

$N =$ **CalculateSimilarity(**e**,** G**);** `// Equation 3.9 or 3.10`

return $rank(N)$**;**



Candidate Entity	Similarity
np_5	a
np_{12}	b
np_3	c
np_x	0

Figure 3.1: The three stages of graph embedding for candidate entities ranking.

3.3.4 Set Expansion using Graph Embedding

Our method starts by taking a seed entity e from the annotator input and returns a ranked list of similar noun phrases (See Algorithm 2). After constructing a graph using the embedding method mentioned in the previous section (See Figure 3.1), we calculate the similarity between the seed entity e and all other noun phrases np_i in the graph G using one of the following similarity methods:

$$Sim^1(np_1, np_2|F) = \frac{\sum_{f \in F} w_{np_1,f} w_{np_2,f}}{\sqrt{\sum_{f \in F} w^2_{np_1,f}} \sqrt{\sum_{f \in F} w^2_{np_2,f}}} \tag{3.9}$$

$$Sim^2(np_1, np_2|F) = \frac{\sum_{f \in F} \min(w_{np_1,f}, w_{np_2,f})}{\sum_{f \in F} \max(w_{np_1,f}, w_{np_2,f})} \tag{3.10}$$

where $Sim(np_1, np_2|F)$ is the similarity between the two noun phrases np_1 and np_2 given the set of features F they have in common, and $w_{(.,.)}$ is the weight of the edge between the noun phrases and features (defined using one of the Equations 3.6-3.8). Equation 3.9 is the cosine similarity and Equation 3.10 is the context-dependent similarity by [120].

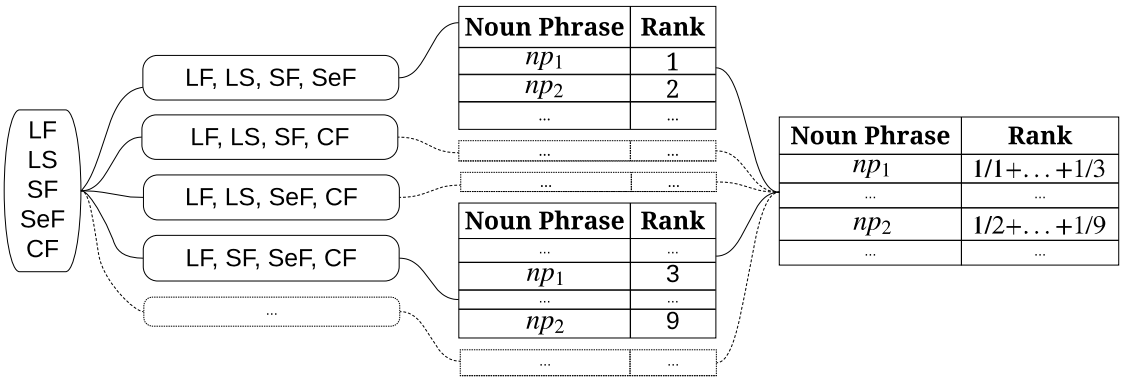


Figure 3.2: Coarse features ensemble method for ranking candidate entities.

3.3.5 Feature Ensemble Ranking

Similar to the use in SetExpan [120], we ensemble the coarse features[4] and rank noun phrases in sublists to reduce the effect of inferior features on the final ranking. The size of each sublist is equal to $|\hat{F}| - 1$, where $\hat{F}$ is the set of all features from Section 3.3.2. We then use the mean reciprocal rank (Equation 3.11) to find the final ranking of each noun phrase using the rank from each sublist (See Figure 3.2).

$$MRR = \frac{1}{|\hat{F}|} \sum_{i=0}^{|\hat{F}|} \frac{1}{rank_i} \tag{3.11}$$

where $rank_i$ is the rank position of the relevant noun phrase for the i-th sublist of features.

3.4 Experiments and Results

In this section, I present our results for evaluating the components of the framework on three publicly available datasets while varying the different parameters of the framework.

Table 3.1: Datasets statistics showing the entity classes, the number of sentences ($|S|$) containing entities, and the number of entities across all sentences.

Dataset Name	**Entity Class**	$\|S\|$ **(with Entities)**	**# Entities**
CoNLL-2003	Location	13519 (36%)	1258
	Person	13519 (31%)	3388
BioCreAtIvE II	Gene	12500 (51%)	10441
GENIA 3.02	Protein Molecule	14838 (55%)	3413
	Cell Type	14838 (27%)	1569
	Virus	14838 (8%)	324

3.4.1 Data Preparation

To test our framework while emulating the full annotation experience of annotators, we use three publicly available gold standard datasets, CoNLL-2003 [127], GENIA [99], and BioCreative II [33], labeled for several entity classes.

We tokenize documents into sentences, then query the emulator (which plays the role of annotators) to label sentences, thus avoiding the required annotation of full documents for better user experience. We require the emulator to label only a single entity class from each dataset, to avoid the complexity of multiple class annotations. Therefore, we created six versions of the datasets, where we keep only one class in each version. Table 3.1 includes some statistics of the datasets. The percentage of sentences with entities of a given class shows the sparsity of those entities.

Table 3.2: Noun phrase extraction performance.

Dataset Name	Entity Class	Accuracy
CoNLL 2003	Location	0.84
	Person	0.76
BioCreAtIvE II	Gene	0.51
GENIA 3.02	Protein Molecule	0.60
	Cell Type	0.24
	Virus	0.50

3.4.2 Noun Phrase Extraction Evaluation

As the ESE method operates at the noun phrase level, the performance of Noun Phrase Extraction (NPEx) influences the overall performance of ESE significantly. Table 3.2 includes the accuracy of NPEx as the percentage of the actual class entities that were extracted as candidate entities (i.e., noun phrases). In the future, ESE performance can be improved by enhancing the performance of candidate entities extraction through, for example, extracting noun phrases formed from typed-dependencies.

3.4.3 Entity Set Expansion (ESE) Evaluation

As ESE requires a seed entity to retrieve the candidate entities and then annotate them in sentences, we tested the influence of seed entity frequency on the method performance. We manually selected two seeds -the most frequent and the least frequent- among all noun phrases (See Table 3.3). Additionally, we varied the weighting measure of the graph edges using one of the three equations (3.6-3.8). Finally, we varied the similarity measure when

Table 3.3: The frequencies of the seed entities (in all sentences) used in the evaluation of the ESE method. One high-frequent and one low-frequent seed entities were used to test the effectiveness of the ranking mechanism per dataset.

Dataset Name	**Entity Class**	**Seed Entity (Count)**
CoNLL-2003	Location	U.S. (296), Washington (26)
	Person	Clinton (70), Von Heesen (1)
BioCreAtIvE II	Gene	insulin (64), GrpE (1)
GENIA 3.02	Protein Molecule	NF-kappa B (552), IL-1RA (1)
	Cell Type	T cells (489), MGC (2)
	Virus	HIV-1 (336), adenovirus E1A (1)

Table 3.4: ESE performance ($p@k$) while using the feature ensemble method. The best performing combination is boldfaced.

		Location			**Person**			**Gene**		
	Eq.#	3.6	3.7	3.8	3.6	3.7	3.8	3.6	3.7	3.8
Seed 1	3.9	0.37	0.40	0.50	0.23	0.23	0.30	0.00	0.03	0.13
	3.10	0.63	**0.73**	0.73	0.03	**0.17**	0.20	0.03	**0.07**	0.07
Seed 2	3.9	0.33	0.33	0.57	0.53	0.40	0.30	0.63	0.63	0.63
	3.10	0.57	**0.70**	0.63	0.47	**0.37**	0.37	0.60	**0.57**	0.60

		Protein			**Cell Type**			**Virus**		
	Eq.#	3.6	3.7	3.8	3.6	3.7	3.8	3.6	3.7	3.8
Seed 1	3.9	0.17	0.23	0.20	0.27	0.50	0.53	0.20	0.13	0.17
	3.10	0.43	**0.43**	0.53	0.17	**0.23**	0.23	0.07	**0.10**	0.07
Seed 2	3.9	0.17	0.60	0.27	0.10	0.20	0.13	0.07	0.03	0.03
	3.10	0.07	**0.30**	0.30	0.07	**0.07**	0.07	0.03	**0.10**	0.03

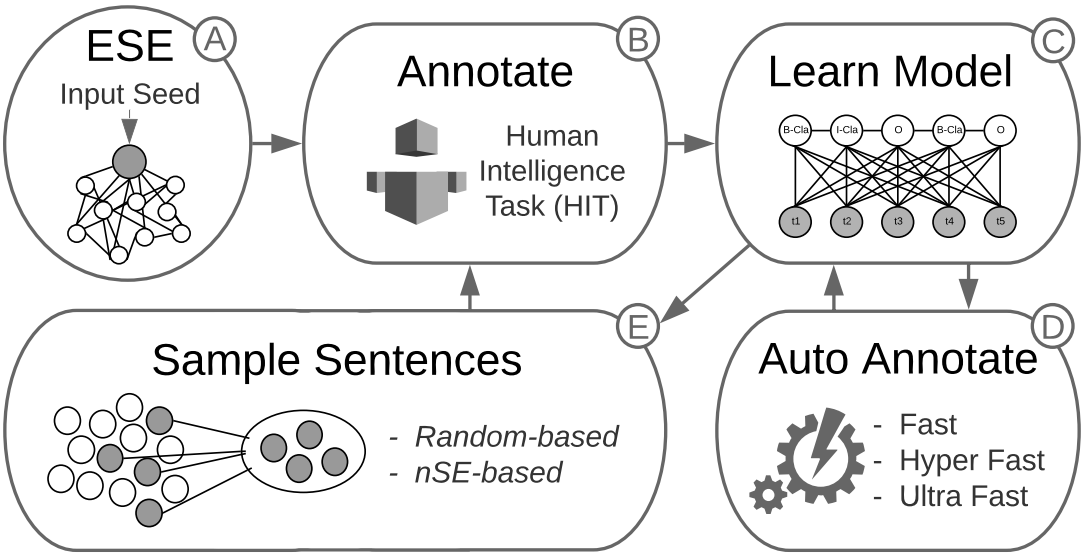


Figure 3.3: End-to-End system pipeline. Arrows represent paths that can be followed to annotate the text.

ranking noun phrases (Equations 3.9 and 3.10).

We tested the performance of ESE with and without using the feature ensemble method from Section 3.3.5. We measured the precision of the method in ranking positive examples of entities similar to the seed entity (Seed 1 and Seed 2) in the top k noun phrases. We designed ESE to output thirty candidate entities (i.e., noun phrases) ranked based on the similarity to the seed term. Therefore, we calculate precision at k ($p@k$), where k is always 30. Table 3.4 shows the best results when using the feature ensemble method, which is more stable than the non-ensemble one (due to the lower standard deviation and non-zero precisions). According to the results, the best combination in terms of the mean and standard deviation is obtained when using TFIDF (Equation 3.7) to weigh the edges and the context-dependent similarity (Equation 3.10) to rank noun phrases.

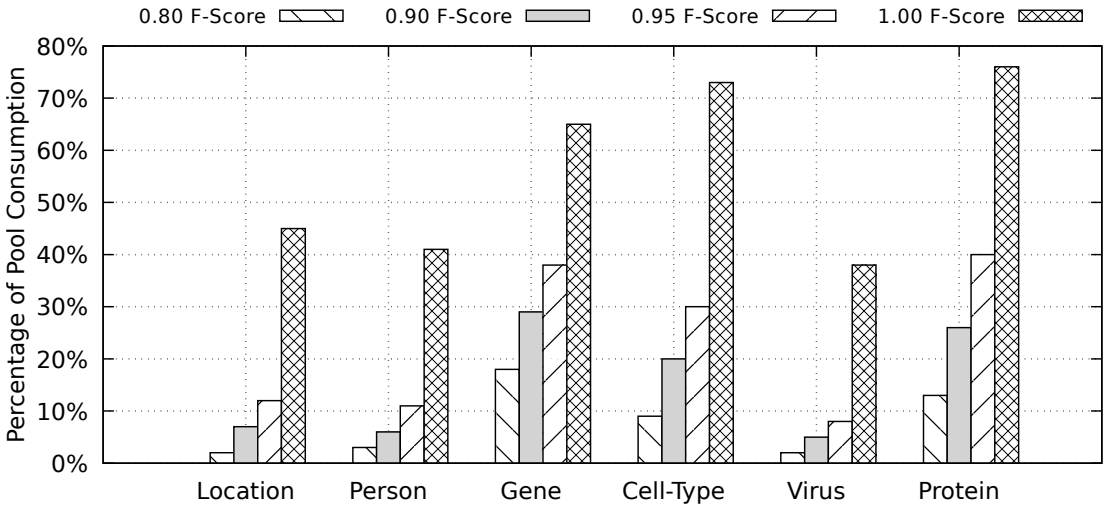


Figure 3.4: Percentage of sentences annotated while using EAL to reach different F-Scores.

3.4.4 Full System Evaluation

We tested our pipeline on three different settings following three different paths in Figure 3.3, where $(\cdot)^*$ is a recurrent path:

1. All Random (AR): $(E, B, C)^*$, where E is a random-based sampler.

2. ESE and AL (EAL): $A, B, C, (E, B, C)^*$, where E is an nSE-based sampler.

3. EAL in addition to auto-annotation (EAA): $A, B, C, (D, C, E, B, C)^*$

We iterate through the loop paths in the starred parentheses $(\cdot)^*$ while successively sampling 100 sentences until we finish all sentences in the pool or reach full F-Score. In Figures 3.5 to 3.7, I show the performance of the first two settings (i.e., AR and EAL) in terms of F-Score. The use of AL and ESE methods always outperformed random sampling. ESE increased the performance of the base model by 35% F-Score on average, allowing us to reach 0.5 F-Score while the random sampler reached only a 0.37 F-Score.

As shown in Figure 3.4, the percentage consumption of the sentence pool to reach up to 0.95 F-Score follows almost a linear growth. However, this cost proliferates if we

[4]Coarse features are the five groups of features in Section 3.3.2 as opposed to the fine ones which are part of each of them.

Table 3.5: Pipeline testing results of the EAL and EAA annotation modes showing the model confidence (σ), F-Scores, and percentage cut from the pool of sentences.

Dataset Name	Entity Class	EAL		EAA Annotation Mode		
		@ 1.0 F		FA	HFA	UFA
		σ	% cut	F-Score (percentage cut)		
CoNLL-2003	Location	0.97	55%	0.99 (46%)	0.93 (83%)	0.82 (91%)
	Person	0.97	59%	0.99 (48%)	0.95 (81%)	0.85 (90%)
BioCreAtIvE II	Gene	0.94	35%	1.00 (35%)	0.96 (50%)	0.89 (69%)
GENIA 3.02	Protein Molecule	0.99	33%	0.98 (36%)	0.87 (71%)	0.74 (85%)
	Cell Type	0.99	62%	0.94 (70%)	0.82 (86%)	0.74 (91%)
	Virus	0.94	24%	0.97 (79%)	0.89 (94%)	0.84 (96%)
Average		0.97	45%	0.98 (52%)	0.90 (78%)	0.81 (87%)

want to reach 1.0 F-Score (needing, on average, around 33% more sentences). Practically speaking, we might want to sacrifice the 5% F-Score improvement to minimize manual labor significantly.[5]

In order to measure the performance of our σ method in comparison with the F1-Score curve (see EAL curves in Figures 3.5 to 3.7), and to compare it with the estimated coverage (EC) method (Equation 3.5), we used the Jensen-Shannon distance metric [67] to measure similarity between the output values of the two methods as in Equation 3.13 below:

$$D_{KL}(P\|Q) = -\sum_i P(i) log(\frac{Q(i)}{P(i)}) \tag{3.12}$$

$$JSD = \sqrt{\frac{D_{KL}(\hat{d}\|m) + D_{KL}(\hat{f}\|m)}{2}} \tag{3.13}$$

[5]This would, therefore, make the annotated data a silver standard instead of a gold standard.

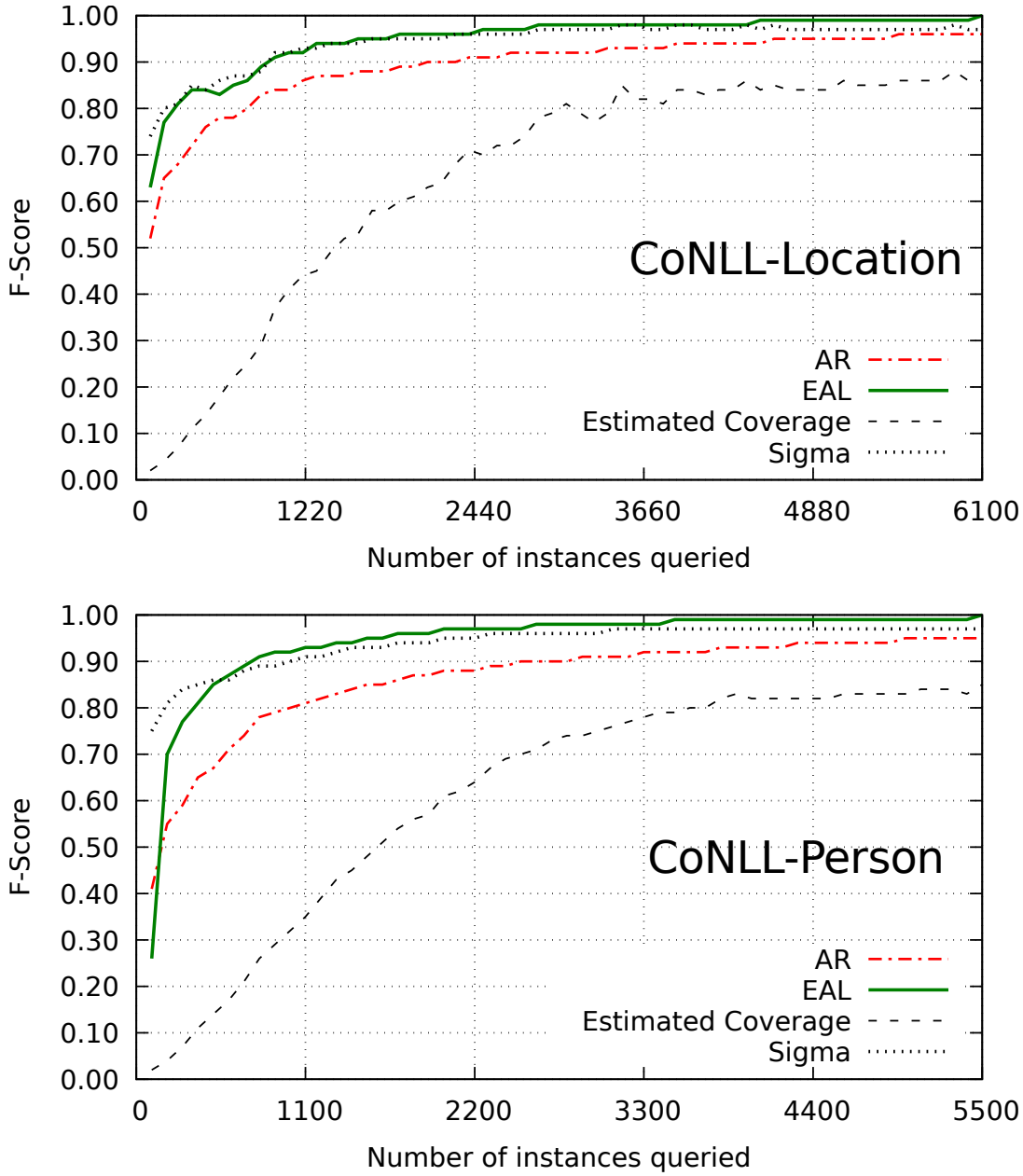


Figure 3.5: Learning curves of the approach on two entity classes from the CoNLL 2003 dataset with different querying strategies. Y-axis value for AR and EAL is the F-Score, and for Sigma confidence (σ) and Estimated Coverage is the value from Equations 3.3 and 3.5, respectively.

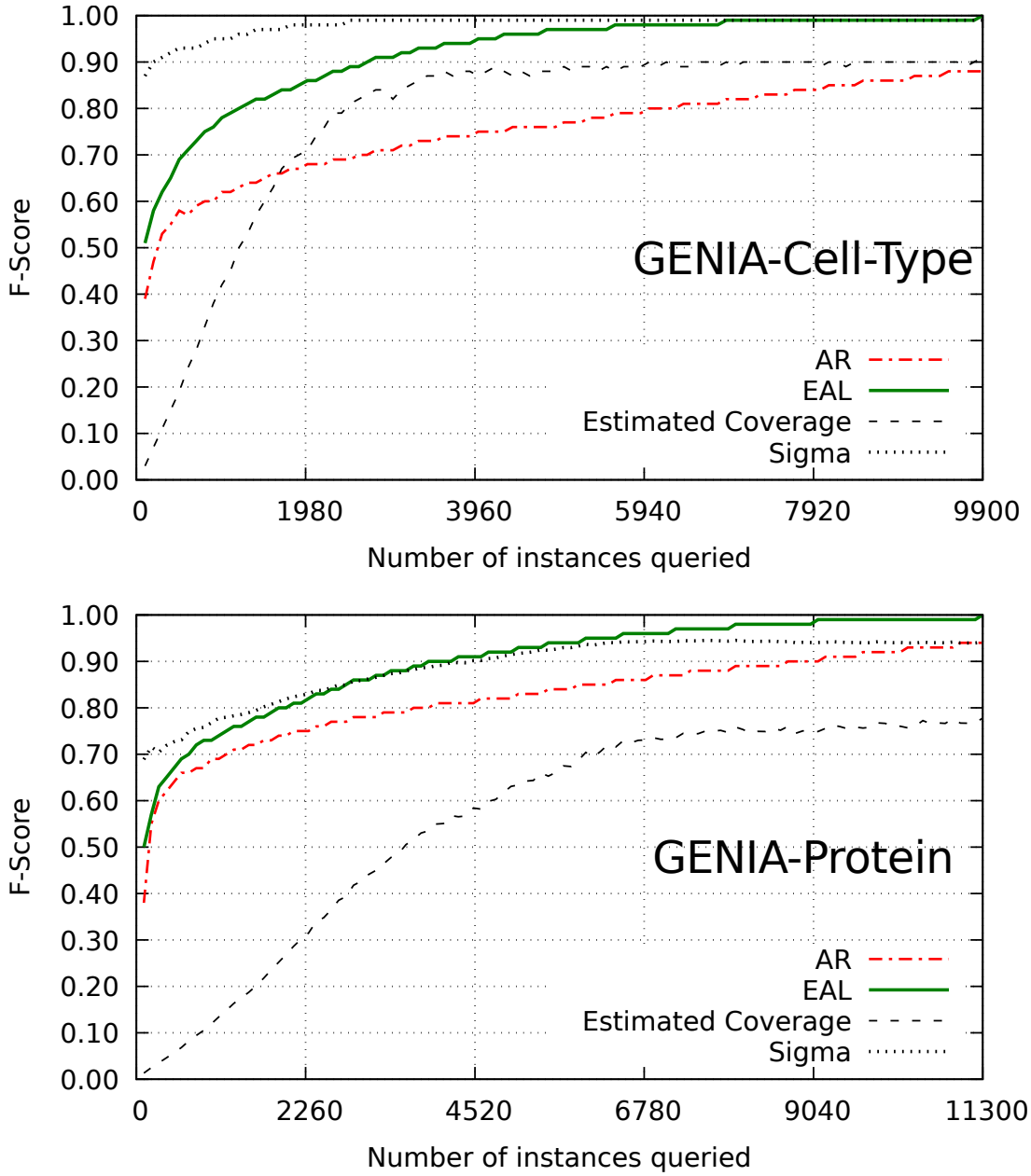


Figure 3.6: Learning curves of the approach on two entity classes from the GENIA dataset with different querying strategies. Y-axis value for AR and EAL is the F-Score, and for Sigma confidence (σ) and Estimated Coverage is the value from Equations 3.3 and 3.5, respectively.

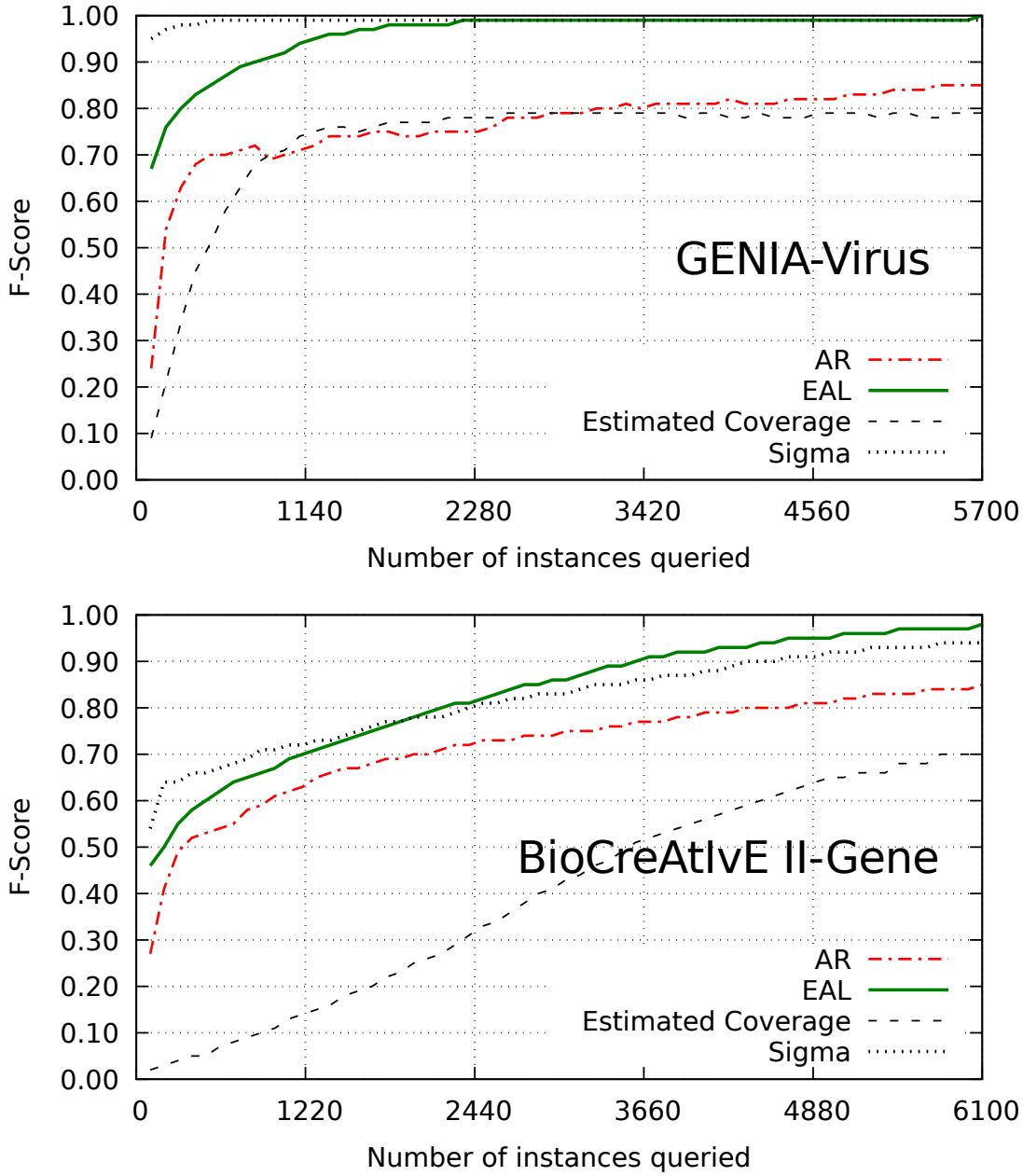


Figure 3.7: Learning curves of the approach on two entity classes one from the GENIA dataset and the second from BioCreAtIvE II with different querying strategies. Y-axis value for AR and EAL is the F-Score, and for Sigma confidence (σ) and Estimated Coverage is the value from Equations 3.3 and 3.5, respectively.

where $\hat{d}$ and $\hat{f}$ are the two distributions (i.e., 1-D arrays of the output values) of σ or EC and F1 scores from all iterations, respectively, m is the pointwise mean of $\hat{d}$ and $\hat{f}$, and D_{KL} is the Kullback-Leibler divergence from Equation 3.12.

Our method σ outperformed the estimated coverage method with a decrease of 96% in dissimilarity, which makes our method more reliable. Additionally, as shown in Table 3.5, on average, σ gives an accurate estimation of the F-Score without labeled data with an error margin of 3% while reaching 1.0 F-Score (i.e., @ 1.0 F). Finally, it is worth mentioning that the σ curves of GENIA-Cell-Type and GENIA-Virus highly overestimated the F-Score curve in the first iterations, which is due to missing entity annotations we found in the gold standard, which mistakenly suggests that we had many false positives. To list a few, GENIA-Cell-Type has missing annotations for "transformed T cells", "transformed cells", and "T cells", where GENIA-Virus have some missing annotations for "HIV-1" and "HIV-2".

Our EAL method also outperformed EC method [128] on the three overlapping entity classes we tested our methods on, from the two datasets Genia and CoNLL-2003. EAL needed 2800, 2900, and 400 fewer sentences to reach the same coverage as [128] for the Location, Person, and Cell-Type datasets, respectively.

Finally, for the last setting, we tested the system using the three auto-annotation modes (i.e., FA, HFA, and UFA) as shown in Table 3.5. While the auto-annotation mode can allow us to reduce up to 87% of the data pool, this drastic saving also reduces the accuracy of the learned model, achieving, on average, around 81% F-Score. Overall, our framework presents a tradeoff between coverage and annotation cost. The HFA auto-annotation mode shows the benefit, especially in a realistic enterprise setting, where the cost to annotate 33% of the data to only increase 10% F-Score (when comparing the average performance of HFA with ESA) is unreasonable.

Table 3.5 appears to show FA being inferior to EAL for the Location class, for example. In reality, FA reduced sentence annotation by 65% to reach 0.99 F-Score. Further, as our

testing criteria demanded that we either reach 1.0 F-Score or finish the sentences, FA tried to finish the pool without performance improvement.

3.5 Related Work

Our work encompasses entity extraction, entity set expansion (ESE), pool-based active learning, and corpora pre-, auto-, and rapid-annotation.

Using a pattern-based ESE (a.k.a., seed set expansion) technique on top of active learning helped our approach in discovering rare patterns and rules that might have been hidden when using only a feature-based system [18]. Our ESE method is similar to [120, 126, 111], where the system starts with a few positive examples of an entity class and tries to extract similar entities. We also use a richer set of features than those in [45, 43, 90] while training a CRF model and to further find positive examples of a given entity class using our ESE method.

Methods such as [112] compute the degree of membership of an entity to a group of predefined entities called seed entities where the class of each group is predefined. Their method does not focus on entity extraction. Rather it focuses on detecting the class of entities during the evaluation step, which makes their method different from what we are trying to achieve here.

Regarding corpus annotation, many notable previous works such as [69] use dictionaries to pre-annotate texts. However, inaccurate pre-annotations can harm the performance and would require an added overhead of modifications and deletions [107]. The techniques in [58] and [128] are AL-based annotation systems. However, their work differs from ours in the following ways: **(1)** we propose a more accurate online evaluation method than theirs, **(2)** we use ESE to bootstrap the learning framework collaboratively with a human-in-the-loop, and **(3)** we provide auto-annotation modes that reduce the number of sentences to be considered for annotation and so allows for better usability of the framework.

3.6 Conclusions, Limitations, and Next Steps

I presented our practical and effective solution to the problem of sparse entity extraction. Consistent with our goal of lowering the cost of annotation (regarding the number of labeled samples and the complexity of labeling) and maintaining the accuracy of NER, our framework builds supervised models and extracts entities with a reduced annotation cost without compromising extraction quality using entity set expansion, active learning, and auto-annotation. Additionally, we provided an online method for evaluating model confidence that enables flexible stopping criteria.

While the technique was able to reduce, on average, around 45% of the data needed to reach about the same accuracy if we consumed all of it, the technique still can benefit from the use of knowledge to further lower the cost of annotating the remaining sentences. In the next chapter, I discuss our knowledge-enabled technique, which exploits online knowledge, rules, and a human-in-the-loop to allow for minimal supervision for NER.

Minimally Supervised Entity Extraction

In this chapter, I introduce our adaptable minimally-supervised technique for entity recognition. While the majority of the NER techniques require domain expertise and extensive manual labor to create labeled data, our approach here reduces that labor with the help of off-the-shelf domain-specific dictionaries and readily available online knowledge sources. The technique starts by extracting candidate entities using these dictionaries. Subsequent processing automatically induces regular expressions with the help of a subject matter expert (SME) as a human-in-the-loop to further expand the candidate entities set. Finally, the system learns entity classifiers to decide which candidate entities are of the desired class, exploiting multi-view lexical, semantic, and contextual features associated with candidate entities. I also present the performance of the technique on six datasets spanning multiple domains (user-generated content, biomedical, and news data). The results show that the pipeline improves recall by 27% while only sacrificing 5% precision, still improving the F1-score by 10% over the NER dictionary-based baseline, thereby justifying the tradeoff.

4.1 Introduction

Numerous practical limitations put a ceiling on the success of NER techniques; some of these techniques suffer from noisy or unavailable annotations (especially for domain-specific entities), due to the high cost, lack of resources and expertise, the time it takes to set-up labeling guidelines [25], the risk of discarding crowd-sourced labels [44], or incom-

plete and weak hand-crafted rules and patterns [1].

Recent state-of-the-art techniques in information extraction use deep neural nets for entity extraction [135]. While these techniques have achieved very good accuracy, they are still limited in two respects. First, they need a considerable amount of data to learn even fairly accurate models (challenging feasibility). Second, they are usually built for generic domains and entity classes (such as people, location, and organization). Specific applications may require new models built from scratch, depending upon new labeled data (challenging reusability). Prior work, such as [118], has attempted to overcome such problems by introducing a dictionary-based, deep neural net entity extraction technique. The technique learns NER models with the help of dictionaries, generic and domain-specific word embeddings, and key phrases from the target corpus. However, the technique remains vulnerable to the vagaries of key phrases and the dictionaries used (see Section 4.5).

I introduce in this chapter our knowledge-enabled, multi-view, minimally supervised, feature-rich entity extraction technique (KnowEx) that addresses the pressing need for more efficient information extraction methods. The design of the method has key practical merit and is consistent with the rising interest in alternatives to costly, fully supervised approaches that do not scale to multiple domains nor preserve IP and data privacy [4]. KnowEx requires off-the-shelf domain-specific dictionaries, readily available online knowledge sources (e.g., WordNet, BabelNet, and pre-trained word embeddings), and minimal input from SMEs.

KnowEx starts by extracting promising candidate entities, employs SMEs-in-the-loop for labeling and filtering, and learns to classify the candidates into valid/invalid entities. Our minimally supervised method exploits rich sets of multi-view features for the candidate entities. Below, I discuss the different aspects of the system and show how our minimally supervised NER solution is a better fit for today's information extraction pipelines in terms of adaptability, growth over time, accuracy, and reduced cost. (The code is available online at https://www.github.com/halolimat/KnowEx).

4.2 Candidate Entity Extraction

Given a sentence such as the following from the BioNLP'13 corpus [93]:

> c-myb is a frequent target of retroviral insertional mutagenesis in murine leukemia virus - induced myeloid leukemia.

A competent English reader who has never before encountered the domain-specific terminology, such as "c-myb", "murine leukemia virus", or "myeloid leukemia", would nevertheless infer that they are concepts of some type. The syntax and the morphological forms of these concepts enable this conclusion [20, 102]. We try to extract all such concepts and label them as candidate entities. We featurize them to learn *what is*, and *what is not* an entity of the desired class combining the lexical, syntactic, and semantic patterns characterizing them.

Next, I discuss the two methods we use to extract candidate entities and the types of features we gather for each candidate. We use the candidates' features to rank-order them for targeted labeling, and to learn a classifier that retains the instances of the desired entity class and ignores the rest.

4.2.1 Language Model-based Extractor (LMEx)

One candidate entity extraction method constructs an n-gram model from the collocations [80] that exist in a *domain-specific dictionary*. An n-gram model determines the set of valid candidate entities $\mathcal{C}$ that appear in the set of sentences $\mathcal{S}$ of a given corpus exploiting the naming regularities captured by dictionaries.

We first filter the dictionary records by removing the digit-only records, records that are of lengths less than two characters, and records overlapping with our stop word list[1]. Then, we tokenize the filtered dictionary records to construct the n-gram model. This

[1]We used the generic English long stop list from https://www.ranks.nl/stopwords

language model-based technique allows us to determine the validity of an n-gram as a candidate entity in a sequence of tokens using a boolean function. A simple segment-based inverted index [65] or exhaustive matching [125] could replace this language model technique. However, such techniques are slower than the n-gram-based technique in Chapter 2 and [5].

More specifically, given a token sequence in a sentence $s_j = \langle t_1, \ldots, t_n \rangle$, where $s_j \in \mathcal{S}$, the goal is to find the valid candidates $\mathcal{C}_j = \{c_1, \ldots, c_m\}$ to be added to the candidates set $\mathcal{C}$, each represented as a pair $c_i = (b, e)$, where b and e are the *begin* and *end* indices of c_i in s_j (i.e., $1 \leq b < e \leq n$), and $\hat{c}_i$ is the surface form of the candidate entity. For example, for the sentences from the BioNLP'13 corpus above, in a dictionary of organisms, "retroviral", "murine", "leukemia", and "virus" are valid unigrams but "c-myb", "mutagenesis", and the rest are not. Similarly, $\hat{c}_i$ ="murine leukemia virus", represented as $c_i = (11, 13)$, is a valid trigram, matching a complete dictionary record and making it an organism candidate entity (i.e., one instance in $\mathcal{C}_j$).

4.2.2 Regular Expression-based Extractor (RegEx)

Dictionaries are incomplete and contribute to low recall. A second candidate entity extraction method addresses this limitation. Automatically induced regular expressions can expand the set $\mathcal{C}$ from above [18]. The implicit sentence structure provided by the descriptive tags of a sentence's tokens, e.g., from part-of-speech (POS) tagging, can be used to extract candidate entities. Previous studies, such as [37] and our method in Chapter 3 and in [4], used POS tag-based regular expressions to extract noun phrases as candidate entities. However, the extraction is done using regular expressions that did not scale well to multiple domains and suffered from low recall due to lack of expressiveness [61]. We address this limitation by automatically inferring regular expressions from a *set* of positive candidate entities by exploiting their POS tag-based patterns, then using these regular expressions (generalizations) for extraction.

For example, in the LaptopReview dataset [104], the POS tags of the aspect terms "hard disk drive" and "1 GB of RAM" are ⟨JJ, NN, NN⟩ and ⟨CD, NN, IN, NNP⟩, respectively. Therefore, regular expressions can be constructed to extract such candidate entities using these POS patterns. Formally, we construct a regular expression from a POS sequence $\langle \texttt{POS}_1, \ldots, \texttt{POS}_n \rangle$ by filling regex templates:

```
(A) X\S+          (B) (?:X\S+\s*)+
(C) \sX\S+        (D) (?:\sX\S+\s*)+
```

where `X` is the first character of a POS tag (e.g., `N` and `V` for `NNP` and `VB`, respectively), `\S` is any non-whitespace characters, `\s` is a whitespace character, and `?:` is a non-capturing group to exclude submatches [124]. `(A)` and `(B)` are templates for the beginning of a regular expression while `(C)` and `(D)` are templates for the rest. `(B)` and `(D)` represent the recurrent POS tags in a given sequence. For example, ⟨NN, NNP⟩ would induce the regular expression `(?:N\S+\s*)+`. As distinct patterns can produce different regular expressions, we construct the union of all extractions and filter the overlapping ones, preferring the longest. Hence, if the extraction of the induced regular expression `N\S+` overlaps with the extractions of `(?:N\S+\s*)+`, we would discard the extraction of `N\S+`.

4.3 Minimal Supervision

The set of candidate entities $\mathcal{C}$ extracted using LMEx and RegEx is noisy (needing filtering) and incomplete (needing expansion). Additionally, the methods leave the challenge of determining *what is*, and *what is not* an instance of a given entity class. Therefore, we need gold labels of a subset of the candidate entities to be able to classify the rest and expand the set of valid entities from $\mathcal{S}$. However, human-centered requirements, such as high-level and minimal inputs from SMEs, are highly desirable in the age of data-hungry AI [7], to reduce the cost of building an accurate candidate classifier. We achieve minimal

supervision by extracting candidates and then: (1) constructing feature-rich embeddings of candidate entities for ranking and classification, (2) requesting SMEs to give only binary valid/invalid labels instead of labeling entity spans in text (see Figure 1.1), and (3) using pre-labeled instances of the other entity classes in the same corpora, to automatically filter out the set $\mathcal{C}$ and improve precision by removing negative candidate instances.

Next, I present the three techniques that underlie our weakly-supervised knowledge-enabled entity extraction method (KnowEx): candidate entity featurization, pattern-based ranking, and partial labels exploitation.

4.3.1 Candidate Entity Featurization

We generally gather two types of features for candidate entities: (1) *Internal Evidence*: The sequences of characters and words that form them, and (2) *External Evidence*: The contexts that surround them. Constructing a semantic model [84] of the candidate entities can, for subsequent semantic processing allow us to rank-order these candidates for SME's input, or for classifying them (see Sections 4.3.2 and 4.4).

Formally, given a candidate c_i in the sentence s_j, we gather the set of features using feature functions $g_{(\cdot)}$ defining its internal and external evidence. Commonly used features include lexical, syntactic, and semantic features as in [4] and Chapter 3. The use of semantic and contextual features on sparse or domain-specific data (such as enterprise data) is not enough and sometimes not available [126]. Therefore, we use diverse features in a complementary manner to collect rich feature sets for effective subsequent processing. For example, the absence of word senses or semantic features for domain-specific entities, such as chemical compounds or auto parts numbers, is compensated by the use of orthographic features.

Using the lexico-orthographic features, we capture the word shape and letter patterns and their abstractions (i.e., short/long word shape (SWS/LWS) features) as follows:

$$g_1(\hat{c}_i) = \texttt{LWS}(\hat{c}_i) \tag{4.1}$$

$$g_2(\hat{c}_i) = \texttt{SWS}(\hat{c}_i) \tag{4.2}$$

where $\texttt{LWS}(\hat{c}_i)$ and $\texttt{SWS}(\hat{c}_i)$ are the long and short (i.e., sans duplicates) abstracts of digits and orthographized letters of $\hat{c}_i$, mapping each letter to the character "L" and each digit to the character "D" and retaining the rest. For example, if $\hat{c}_i$ ="ABC-123", the long word shape $\texttt{LWS}(\hat{c}_i)$ ="LLL-DDD" and the short word shape $\texttt{SWS}(\hat{c}_i)$ ="L-D".

To exploit the syntactic patterns, we record the POS tags and the explicit lexical patterns surrounding the candidate entities:

$$g_3(c_i, s_j) = \texttt{LR-POS}(c_i, s_j) \tag{4.3}$$

$$g_4(c_i, s_j) = \texttt{LR-LEX}(c_i, s_j) \tag{4.4}$$

where LR-POS and LR-LEX record the left-right (LR) contextual POS tags (i.e., $\langle \mathrm{p}_{b-1}, \mathrm{p}_{e+1} \rangle$) and tokens (i.e., $\langle t_{b-1}, t_{e+1} \rangle$), respectively.

To abstract from the word-order information while describing the universal dependencies of c_i, we employ syntactic dependency parsing using Spacy[2] as follows:

$$g_5(c_i, s_j) = \texttt{DEP}(c_i, s_j) \tag{4.5}$$

where DEP extracts a set of patterns from dependency arcs, describing the syntactic relations (i.e., the head, lefts, or rights) c_i has with some other tokens in s_j. For example,

[2] spacy.io/usage/linguistic-features

$\{\langle \texttt{amod}, \texttt{J} \rangle, \dots\}$ where amod is the arc label and J is the first character of the POS tag of the other token in the relation.

To capture the word semantics for each candidate entity, we use WordNet [88], BabelNet [92], and BERT contextual-embeddings [27]:

$$g_6(\hat{c}_i) = \texttt{WordNet}(\hat{c}_i) \tag{4.6}$$

$$g_7(\hat{c}_i) = \texttt{BabelNet}(\hat{c}_i) \tag{4.7}$$

$$g_8(c_i, s_j) = \texttt{BERT}(c_i, s_j) \tag{4.8}$$

where g_6 and g_7 returns the unordered sets of senses and Wikipedia categories (such as "Rare_diseases", "Digestive_system", or "Urban_animals") of $\hat{c}_i$, respectively. g_8, as the bottom-up distributional semantics, returns the average vector of all the contextual embeddings of c_i's tokens in the sentence s_j.

Finally, we consolidate and vectorize the features of the set $\mathcal{C}$ of all candidate entities in sentences, creating three feature views:

1. **Lexical, Syntactic, and Semantic View (lsv):** To capture the syntax and semantics of a candidate entity in a sentence, we get the set of features $\texttt{L}_{(\cdot)} \subseteq \texttt{L}$ from its lexico-orthographic (4.1-4.2), lexico-syntactic (4.3-4.4), syntactic (4.5), and word senses (4.6), where L is the set of all the features of $c_{(\cdot)} \in \mathcal{C}$. Then, for a $c_{(\cdot)}$ in s_j, we create a binary vector $v^{lsv}_{(\cdot)} \in \{0,1\}^{|\texttt{L}|}$.

2. **Wikipedia Categories View (wcv):** Given a domain-specific dictionary D, we get the set $\texttt{W}_r$ containing the Wikipedia categories of a dictionary record r using (4.7). Then, we create a binary vector $v^{wcv}_{(\cdot)} \in \{0,1\}^{|\texttt{W}|}$ for each $\hat{c}_{(\cdot)}$, where $\texttt{W} = \bigcup_{r \in D} \texttt{W}_r$ is the set of all such Wikipedia categories of the dictionary records. Using the categories of the dictionary records as a reference for vectorizing the candidate entities allows us

to maintain some degree of relatedness to the subject area inherent in the dictionary records.

3. **Contextual Embeddings View (cev):** For each c_i in s_j, we construct the view v_i^{cev} by running s_j through BERT's pre-trained transformer architecture (4.8) and concatenating the output of the last four hidden layers per token. Formally, $v_i^{cev} \in \mathbb{R}^{|\mathtt{D}|}$, where $\mathtt{D}$ is the dimension of the embedding.

4.3.2 Pattern-based Candidate Entity Ranking

To fully exploit and minimize SME input, we show them only a subset of candidate entities that have the greatest potential to improve the overall coverage/recall instead of labeling each occurrence [78, 134]. Therefore, given a set of rank-ordered candidate entities, a SME labels the top k candidates as *valid* or *invalid* instances of the targeted entity class (cf. relevance feedback). We expect the top k candidates (e.g., 50) to characterize the majority class (via being either most probably negative or most probably positive).

We rank order the candidate entities $\mathcal{C}$ using a linear model of their aggregated feature weights, as I discuss below.

The ultimate goal of our ranking procedure is to surface candidate entities that are semantically related [14] to the target class. The significance of a feature f of a candidate entity for class membership is determined by the frequency of occurrence of the feature in related entities of the same class, thereby measuring their similarity. Formally, given the set of candidate entities $\mathcal{C}$, we consider the frequency of a feature f from lsv[3] as the feature's weight: $w_{f,\mathcal{C}} = count(f)$, where $f \in \mathtt{L}$ and $count(\cdot)$ is the frequency over all candidates in $\mathcal{C}$. To score each $c_{(\cdot)}$, we use the aggregated weights as follows:

$$Score_{lsv}(\hat{c}_i) = \sum_{s_j \in S} \sum_{\hat{c}_i \equiv c_i \in C_j} w_{\mathtt{L}}^T \cdot v_i^{lsv} \tag{4.9}$$

[3]The rich lsv view was superior to the sparse wcv.

where $w_{\mathtt{L}}^T$ is the transposed vector of weights of the features in $\mathtt{L}$. This linear scoring model surfaces the top candidates that exhibit the majority class features (i.e., the most frequent ones). Therefore, labeling them would allow us to learn the categorization of many instances.

4.3.3 Partial Labels Exploitation

As we extract and label instances of a single entity class at once (e.g., Locations, Organisms, or Diseases), we are interested in a further minimization of false positives in the candidates set $\mathcal{C}$. For that, we consider other (semantically disjoint) labeled entity classes in a given corpus as negative examples for the current class at hand. Formally, given a set of sentences $\mathcal{S}$ which contains entities from the set of classes $\mathcal{E}$ (e.g., {Location, Organization, Person}), we call $\mathcal{S}$ a partially labeled corpus if the current labeled classes $\mathcal{L} \subset \mathcal{E}$, e.g., $\mathcal{L} = \{\text{Location, Person}\}$. Given $\mathcal{S}$, the set $\mathcal{C}$ of the instances of targeted class $\mathcal{T}$ (e.g., Organization), where $\mathcal{T} \in \mathcal{E} - \mathcal{L}$, can be filtered to improve the precision using instances from $\mathcal{L}$, as negative examples for $\mathcal{T}$— as the instances in $\mathcal{C}$ that might (erroneously) be from the negative classes $\mathcal{L}$.

4.4 Multiview-based Classification

After extracting candidate entities, featurizing them, and acquiring for some candidates their valid/invalid category labels from a SME, we train a classifier to learn to categorize the rest of the candidates (for which we lack gold labels), to construct the final set of entities.

Consider the two concepts “murine leukemia virus” and “myeloid leukemia” from Section 4.2 labeled by a SME, the first as an instance of the *Organisms* entity class and the second as a *Non-organisms* mention (an instance of the negative class *Cancer*). We want to

learn the set of features that represent the positive class (e.g., $g_7(\cdot)$ ="Gammaretroviruses") and the negative class (e.g., the dependency feature $g_5(\cdot) = \langle \texttt{amod}, \texttt{V} \rangle$ from the syntactic relation between "myeloid leukemia" and "induced"). Similarly, after extracting candidate entities, featurizing them, and acquiring valid/invalid category labels for some of them from a SME, we train a classifier to categorize the remaining candidates (lacking gold labels), to construct the final set of entities.

Binary classification and voting using the multi-views: Given a set of candidate entities $\mathcal{C}$, we learn three classifiers, one per each feature view from Section 4.3.1 using their feature vectors (i.e., v^{lsv}, v^{wcv}, and v^{cev}). As each classifier can now learn and independently predict a valid/invalid entity instance using its feature view, we designed a voting mechanism to obtain consensus labels using majority voting rule. Formally:

$$l = \begin{cases} 0, & \text{if } \sum_{q \in V} \sigma(v^q) \leq 1 \\ 1, & \text{otherwise} \end{cases} \tag{4.10}$$

where $V = \{lsv, wcv, cev\}$ and $\sigma(v^q)$ is the binary classification result on the feature view v^q.

Partial labels as negative/invalid instances: To expand the set of invalid instances obtained from a SME input, we featurize and add all of instances of the negative classes from $\mathcal{L}$ (see Section 4.3.3) to the set of gold labels for training the classifiers.

Data Imbalance: To partially overcome the effect of data imbalance, we use oversampling [51, 16] and undersampling [77] techniques. SMOTE [16], for example, oversamples the minority class by adding synthetic samples based on feature-space similarities among the existing instances in that class.

4.5 Experiments

This section answers the following questions. First, how does KnowEx perform on benchmark datasets in comparison with other NER techniques (both supervised and dictionary-based)? Second, how effective was the idea of regular expressions induction for improving recall? Third, how effective was the use of multi-view features and knowledge? Fourth, how much cost reduction or accuracy improvement was our approach able to provide in comparison with other techniques?

Figure 4.1 shows the KnowEx pipeline, including the candidate entity extractors (LMEx and RegEx), the multi-view features, the classifiers, and the SME-in-the-loop. The pipeline initially extracts a set of candidates using LMEx. The candidates are then featurized using our multi-view featurizers. We then rank order the candidate entities based on the frequency of their features (assuming that LMEx extractions are high precision but low recall) and query the SMEs-in-the-loop to label them as valid/invalid instances. The pipeline then passes these labeled candidates to RegEx to automatically induce regular expressions (to expand the set of candidates further). After doing the same for RegEx extractions, we pass the labeled instances to learn a binary classifier (to classify the unlabeled candidates automatically).

Below, I present the details about the experimental settings (listing the dataset we evaluated the system on, and the knowledge we used to allow for minimal supervision). Then, I present the evaluation of each component of the pipeline (to highlight the strengths and weaknesses of each component and how each component is contributing to the overall performance of the pipeline) and compare our pipeline with other NER techniques.

4.5.1 Experimental Settings

I compare our method with our base-line and the multi-domain dictionary-based NER method AutoNER [118]. To highlight the trade-off between cost and accuracy, I also show

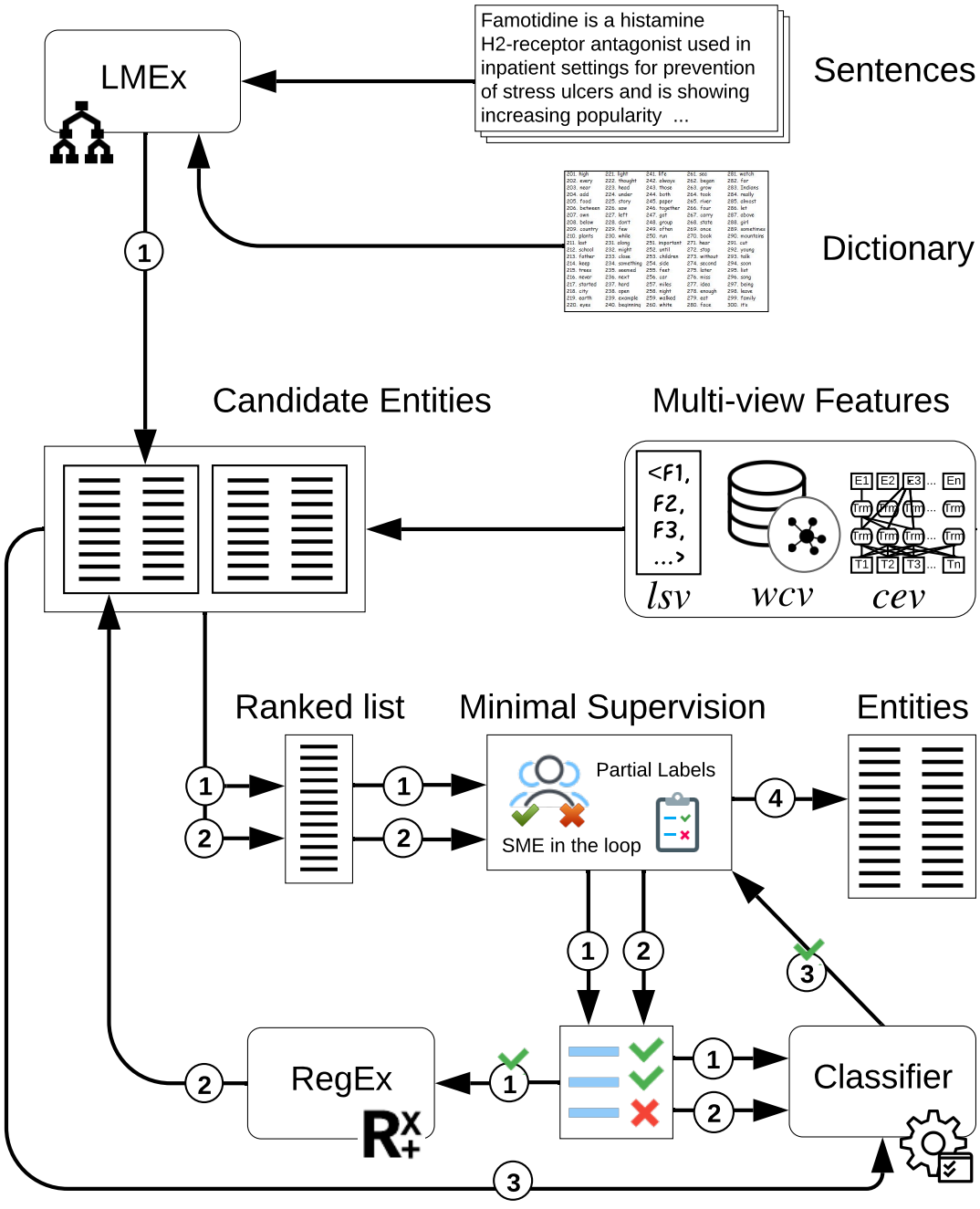


Figure 4.1: The system architecture of our adaptive knowledge-enabled pipeline (KnowEx).

Table 4.1: Datasets statistics: The number of sentences and entities in each corpus (C# 1 to 10).

			Train Data		Test Data	
C#	Dataset Name	Entity Class	# Sentences	# Entities	# Sentences	# Entities
1	Laptop Review	AspectTerm	2436	801	800	393
2	NCBI	Disease	5432	1577	940	403
3	BC5CDR	Chemical	4560	1007	4797	1020
4		Disease	4560	1384	4797	1337
5	MIT Restaurent	Restaurant Name	7660	1028	1521	296
6		Cuisine	7660	475	1521	179
7	CONLL 2003	Location	14041	1105	3250	426
8		Organization	14041	2295	3250	634
9	BioNLP13CG	Organ	3033	58	1906	55
10		Organisms	3033	128	1906	101

the results of the best performing supervised technique [96] across multiple domains, i.e., Flair [3].

Datasets: To examine the effectiveness and generalizability of the different NER methods on English text, we evaluate them on ten entity classes from three domains (see Table 4.1). All methods are tested using the test datasets, while the train datasets were only used to train the supervised technique, Flair. Following are the details domains of the datasets:

1. **Crowd Sourced or Transcribed**:

   - **Laptop Review** [104]: This dataset is from the SemEval-2014 Task 4[4], which was extracted from customer reviews of laptops. The data is annotated for aspect terms as entities (e.g., “performance”, “build quality”, and “warranty”).

[4] http://alt.qcri.org/semeval2014/task4/

- **MIT Restaurant** [72]: This dataset is tagged for multiple entity classes in BIO format. We use the restaurant name and cuisine classes in our experiments (e.g., "Burger King" and "Steak House").

2. **Medical Domain**:

   - **BioCreative V Chemical Disease corpus (BC5CDR)** [131]: The corpus of this dataset is from PubMed articles annotated for chemicals, diseases, and their interactions. In our experiments, we use the chemical and disease entity classes (e.g., "Calcitriol" and "hypercalcemia").
   - **NCBI** [29]: The corpus of this dataset is taken from PubMed abstracts and was annotated for disease names (e.g., "hereditary ovarian cancer").
   - **BioNLP'13** [93]: This dataset is from the BioNLP Shared Task (BioNLP-ST) 2013[5] annotated originally using the standoff format. We use the organ and organism entity classes in our experiments (e.g., "kidney" and "murine").

3. **News Domain**:

   - **CoNLL 2003** [127]: The dataset was originally taken from a collection of news wire articles from the Reuters Corpus. We use two out of the four annotated entity classes in the dataset, locations and organizations (e.g., "Jordan" and "U.N.").

For AutoNER [118], we formatted the input data as required using their Tie or Break scheme, used AutoPhrase [117] to extract key phrases, and used the same dictionaries we tested our method (see details below about the dictionaries used). For the biomedical domain, we used the same embeddings as in [105], and for the other domains, we used GloVe [101].

[5] http://2013.bionlp-st.org/

Domain-specific Dictionaries: We used the Aspect Term, Chemical, and Disease dictionaries from AutoNER. The rest of the dictionaries were either taken off-the-shelf or scraped from online sources (e.g., Wikipedia). We used the following dictionaries:

- **Fast Food Restaurants** [47]: This dictionary has 548 unique names of fast restaurants in USA taken from a data science class materials at UBC.
- **Cuisine** [49]: This dictionary was built automatically from Zagat.com and contains 145 unique cuisine names.
- **World Cities and Countries** [11]: This dictionary contains 23809 unique city names with above 15,000 inhabitants extracted from geonames.
- **Organizations** [100]: This dictionary contains 9015 unique organization names taken from the FreeLing project codebase.
- **Organs** [59]: This dictionary contains 66 unique organ names scraped from online.
- **Organisms dictionary** [133]: This dictionary contains 395 unique organisms names scraped from Wikipedia.

For AutoNER, we used AutoPhrase [117] to extract key phrases to build dictionaries from the data (i.e., Full Dictionary). As for the core dictionaries that AutoNER requires, we used the same domain-specific dictionaries mentioned above.

Pre-trained Word Embeddings: We use BERT's contextual word embeddings provided by [36]. For AutoNER in the biomedical domain, we used the same embeddings as in [105], and for the other domains, we used GloVe [101].

Domain-specific Knowledge: We used BabelNet indices version 4.0.1 ($\sim$ 29Gb) and their API[6] to query the indices for the Wikipedia categories of each candidate entity. As

[6] github.com/marcevrard/BabelNet-API

Table 4.2: The aggregated weights of Wikipedia categories for the domain-specific dictionaries.

C#	Wikipedia Category (Weight)
1	American_inventions (19), Computer_storage_devices (9), USB (5)
2	Rare_diseases (96), Syndromes (22), Rare_cancers (8)
3	Hepatotoxins (48), Alcohols (36), Phenol_ethers (30)
4	Psychiatric_diagnosis (65), Medical_emergencies (40), Pain (17)
5	Fast-food_franchises (92), Yum!_Brands (8), Kentucky_cuisine (3)
6	American_cuisine (8), Mediterranean_cuisine (8), Street_food (5)
7	Cities_in_Ohio (49), Holy_cities (41), City-states (33)
8	German_brands (18), Central_banks (15), Serie_A_clubs (12)
9	Digestive_system (12), Exocrine_system (9), Endocrine_system (8)
10	Urban_animals (24), Scavengers (17), Livestock (13)

in Section 4.3.1, given a domain-specific dictionary D, we create the Wikipedia categories view (wcv). Using the categories of the dictionary records as a reference for vectorizing the candidate entities would allow us to maintain a certain degree of relatedness to the subject area inherent in the dictionary records (see examples of the top categories and their weights in Table 4.2).

Metrics: To measure the performance of the binary classifier, we use macro-averaged scores, giving equal weights to all the classes, since our candidate entities sets are imbalanced [79]. As for evaluating candidate entity extraction and the final set of extracted entities, we use precision, recall, and F1-score for a given entity class from each of the sentences pools separately.

Table 4.3: Performance of the dictionary-based entity extractor (LMEx) while using raw and filtered dictionaries. Bold-faced is for the improved precision while the underlined show the ones without a difference.

	Raw LMEx			Filtered LMEx		
C#	P	R	F1	P	R	F1
1	0.77	0.53	0.63	**0.79**	0.51	0.62
2	0.69	0.60	0.64	**0.82**	0.59	0.69
3	0.91	0.59	0.71	**0.93**	0.58	0.71
4	<u>0.88</u>	0.64	0.74	<u>0.88</u>	0.64	0.74
5	<u>0.89</u>	0.23	0.37	<u>0.89</u>	0.23	0.37
6	<u>0.67</u>	0.61	0.63	<u>0.67</u>	0.61	0.63
7	0.38	0.75	0.50	**0.54**	0.76	0.63
8	0.58	0.66	0.62	**0.65**	0.65	0.65
9	<u>0.49</u>	0.37	0.42	<u>0.49</u>	0.37	0.42
10	<u>0.84</u>	0.29	0.43	<u>0.84</u>	0.29	0.43

Table 4.4: Performance of the dictionary-based entity extractor (LMEx) with filtered dictionaries vs. AutoNER. Bold-faced values reflect the best performances.

	LMEx			AutoNER		
C#	P	R	F1	P	R	F1
1	0.79	0.51	**0.62**	0.71	0.47	0.57
2	0.82	0.59	**0.69**	0.77	0.57	0.66
3	0.93	0.58	**0.71**	0.91	0.55	0.68
4	0.88	0.64	**0.74**	0.83	0.61	0.70
5	0.89	0.23	**0.37**	0.74	0.20	0.32
6	0.67	0.61	**0.63**	0.65	0.59	0.62
7	0.54	0.76	**0.63**	0.55	0.65	0.60
8	0.65	0.65	**0.65**	0.68	0.50	0.58
9	0.49	0.37	**0.42**	0.35	0.37	0.36
10	0.84	0.29	**0.43**	0.82	0.28	0.42

4.5.2 Performance Evaluation

In this section, I separately present evaluation results for our pipeline on each pool of sentences. The SMEs-in-the-loop are emulated using gold annotations from each public benchmark mentioned above.

LMEx and RegEx evaluation: As in Section 4.2.1, filtering the domain-specific dictionaries to build language models for LMEx improved precision for 5 datasets while the other 5 had no change (see Table 4.3). The maximum improvement in precision was 16% for the CoNLL-2003 Location dataset. Ultimately, filtering dictionaries improved average precision by 4 points, improving F1 from 0.57 to 0.59.

In Table 4.4, our baseline language model-based extractor (LMEx)[7] beats AutoNER on all datasets in terms of precision, recall, and F1, except for precision on corpora 7 and 8. However, as with any dictionary-based extraction method, they both suffer from low recall. The break-and-tie scheme of AutoNER fails to improve precision significantly, and the technique is still prone to dictionary incompleteness. Hence, the need for our complete pipeline without the high cost associated with supervised methods.

As for RegEx, it was able to improve recall by an average of 0.19, with a minimum of 0.05 for the BC5CDR-disease class and a max of 0.42 for the organisms class.

Candidate ranking and SME labels evaluation: As mentioned before, we featurize and rank the extractions of LMEx and RegEx in two steps using the best performing ranking mechanism with which we experimented (i.e., lsv). First, we rank the extractions of LMEx and query SMEs to label the top 50 of them. The precision at k (p@k) of this ranked list, on average, was around 0.88 (i.e., 44/55 were valid entities, see Table 4.5). Additionally, over all datasets, almost 10% of the candidate entities were eliminated by filtering the

[7] For the remaining parts of this chapter, I use LMEx to refer to the method while using *filtered* dictionaries

Table 4.5: Precision at k (P@k) and the number of instances covered in sentences (Instance Count) for the best performing ranking mechanism (lsv) on LMEx extractions. SMEs-in-the-loop label at most 50 instances.

C#	k	P@k	Inst. Count
1	50	0.94	218
2	50	0.94	349
3	50	1.00	1350
4	50	0.98	1132
5	32	0.88	93
6	50	0.90	303
7	50	0.80	709
8	50	0.74	254
9	18	0.78	55
10	8	0.88	132

Table 4.6: The number of candidate entities extracted using LMEx (with and without filtering) and the combined extractions of LMEx with filtered dictionaries and RegEx.

	Candidates Count		
C#	Raw LMEx	Filtered LMEx	Filtered LMEx + RegEx
1	482	456	3030
2	842	707	7582
3	3517	3399	52347
4	3287	3282	50025
5	106	106	3847
6	485	485	4797
7	3550	2588	16102
8	1507	1356	18562
9	118	118	16970
10	179	179	21131

Table 4.7: The number of labeled candidates by a SME and the precision of labels (P-Valid and P-Invalid).

	SME Labels					
C#	# Valid	# Invalid	# Unlabeled	% Unlabeled	P-Valid	P-Invalid
1	315	657	2058	0.68	0.71	1
2	471	1118	5993	0.79	0.78	1
3	1552	8764	42031	0.80	0.97	1
4	1377	8177	40471	0.81	0.87	1
5	95	1004	2748	0.71	0.98	1
6	582	1834	2381	0.50	0.54	1
7	960	2148	12994	0.81	0.88	1
8	363	2275	15924	0.86	0.71	1
9	192	3610	13168	0.78	0.40	1
10	409	4676	16046	0.76	0.73	1

dictionaries used by LMEx (see Table 4.6). Second, we query SMEs to label the next batch extracted using the induced regular expressions from the first 50. This step allows us to filter out invalid entities and reduce the noise in RegEx extractions. RegEx expanded the set of candidates by 93% across all datasets (from 12676 to 194393 candidates).

On average, the precision of the 100 SMEs' binary labels was around 0.76 for the valid entities and 1.0 for the invalid ones (see Table 4.7). The lost precision of SME labels is due to our minimally supervised, context agnostic labeling technique. This design choice to minimize supervision allows the SME to label surface forms instead of spans of texts without looking at the context of the mentions. This allowed the SME to label 50 candidate entities that corresponded to 1099 instances with a precision of 0.98 and 1.0 for the valid and invalid classes, respectively, in the case of corpora 5. This excellent performance was because some domains are more suitable to reliance on internal evidence, which is what the SME is exploiting while giving binary labels (e.g., see the precision for corpora 3, 4, 5, and 7). Notably, all labels account for only around 25% of the candidates, leaving the rest

Table 4.8: The precision (P), Recall (R), and F1 of the Micro- and Macro-averaged candidate entity classifier scores.

	Micro Averaged			Macro Averaged			
C#	P	R	F1	P	R	F1	Support
1	0.86	0.86	0.86	0.73	0.80	0.76	2058
2	0.86	0.86	0.86	0.68	0.89	0.77	42031
3	0.86	0.86	0.86	0.60	0.80	0.69	40471
4	0.81	0.81	0.81	0.61	0.76	0.68	5993
5	0.91	0.91	0.91	0.58	0.74	0.65	2748
6	0.77	0.77	0.77	0.57	0.68	0.62	2381
7	0.96	0.96	0.96	0.89	0.86	0.87	12994
8	0.72	0.72	0.72	0.58	0.82	0.68	15924
9	0.99	0.99	0.99	0.64	0.80	0.71	13168
10	0.99	0.99	0.99	0.78	0.61	0.68	16046
Average	0.87	0.87	0.87	0.67	0.78	0.72	15381

to be classified using our binary classifier to be built next.

Candidate entity classification: Using the labeled set of 100 candidates, we learn binary classifiers for the valid/invalid categorization of candidates. To avoid over-fitting, we remove the features that have a document frequency of less than a threshold of 0.01. Finally, we train Multi-layer Perceptron classifiers [42] with the following best-performing configurations: the LBFGS method [70] to optimize the loss-function, two hidden layers of sizes 500 and 200, and a batch size of 10.

Table 4.8 contains the micro and macro-averaged scores measuring various performance metrics of each binary classifier on the candidate entity classification task to automatically label the 75% of the candidate entities without gold labels. The results show the best performing balancing technique (SMOTE).

The macro-averaged F1 score of classifiers across all sentences was, on average, 0.72.

Table 4.9: Pipeline performance after using the classifier on top of LMEx and RegEx methods, and with partial labels on the test corpora (C# from 1 to 10). Absent partial labels are replaced with strikethrough text.

	Classifier Extractions			Partial-1			Partial-2		
C#	P	R	F1	P	R	F1	P	R	F1
1	0.64	0.67	0.66	~~0.64~~	~~0.67~~	~~0.66~~	~~0.64~~	~~0.67~~	~~0.66~~
2	0.44	0.65	0.52	~~0.44~~	~~0.65~~	~~0.52~~	~~0.44~~	~~0.65~~	~~0.52~~
3	0.45	0.85	0.59	0.52	0.85	0.64	~~0.52~~	~~0.85~~	~~0.64~~
4	0.28	0.71	0.41	0.37	0.71	0.49	~~0.37~~	~~0.71~~	~~0.49~~
5	0.42	0.37	0.39	0.44	0.37	0.40	0.48	0.37	0.42
6	0.33	0.75	0.46	0.42	0.75	0.53	0.53	0.75	0.62
7	0.78	0.84	0.81	0.93	0.84	0.88	0.96	0.84	0.89
8	0.19	0.81	0.30	0.21	0.81	0.34	0.41	0.81	0.55
9	0.31	0.60	0.41	0.31	0.60	0.41	0.42	0.60	0.49
10	0.72	0.69	0.71	0.72	0.69	0.71	0.73	0.69	0.71

However, in terms of the average micro-averaged F1, the performance of the classifier was around 0.87. The difference between the macro and micro scores were due to the severe class imbalance problem that we partially solved using SMOTE. The proportional frequency of the positive class to the negative class was around 1:19 (in the worst case) and 1:2 (in the best case). Nevertheless, the classifier improved the precision of the candidates' pool by 0.39 while sacrificing only 0.02 recall with an overall increase in F1 score on average by 0.40. This performance was due to a 10 out of 10 improvement in terms of precision, 4 out of 10 in terms of recall, and 10 out of 10 in terms of F1 over all datasets.

Table 4.9 lists the performance of the pipeline that relies on the classifier to decide the final set of entities in all sentences. As we were initially motivated to improve the recall of our baseline (LMEx), the table shows a significant improvement of around 0.17 points (on average) over LMEx. This performance was due to a 10 out of 10 improvement in recall but with an average sacrifice of 0.29 in precision (i.e., decreasing precision for 9 out

of 10 datasets). Nevertheless, at this stage, our technique exhibits inherent precision-recall trade-off and provides a general solution that scales to multiple domains using off-the-shelf dictionaries and generic knowledge with minimal supervision. Next, I will discuss the use of partial labels and SMEs-in-the-loop to improve precision while maintaining the improved recall.

Partial Labels and SMEs-in-the-loop: While maintaining our commitment to weak supervision, we further improved the resulting precision using partial labels (see Section 4.3.3) and minimal SME input. In Table 4.9, as expected, the use of partial labels improved the average F1-score from 0.53 to 0.60 (with a max of 0.22 increase in precision). This performance was due to an improvement between 0.09 to 0.22 for 8 out of the 10 datasets in terms of precision.

After rank-ordering the pipeline extractions following the filtering by partial labels, we query the SME-in-the-loop for binary labels to improve the overall performance of the pipeline. The SME's job at this stage is to filter out the false positives (among the top 50-250) to improve precision. Ultimately, our system obtained (on average) a precision, recall, and F1-score of 0.70, 0.68, and 0.67, respectively. By labeling only 50 instances, we were able to obtain (on average) a percentage improvement of 7.3% in precision, and by using 250 labels, the percentage improvement was around 27%. Finally, we obtained an F1 score of 0.67, suggesting that having SMEs-in-the-loop providing only minimal input significantly improved overall performance.

Table 4.10 contains the final results in one place. As listed in the table, using only available online knowledge and 350 binary labels from the SMEs-in-the-loop[8], our full pipeline was able to significantly improve the performance of the NER in comparison with other dictionary-based techniques (i.e., LMEx and AutoNER). The full pipeline was able to, on average, maintain similar precision to AutoNER but with an increase in the recall by 0.20,

[8]Recall, if each label takes, on average, a generous 10 seconds, the full annotations will take close to one-hour in comparison with 25 hours only for reading, see Section 1.1.3

Table 4.10: Multi-domain NER performance comparison (Corpus # and precision/recall/F1 score). Gold data are labeled with IOB formatted labels. A SME-in-the-loop for KnowEx provides minimal supervision.

Method	Human Effort / Knowledge	\C#	1	2	3	4	5	6	7	8	9	10
Flair	Gold Labels + Flair Embeddings	P	0.84	0.85	0.88	0.80	0.89	0.87	0.91	0.97	0.73	0.88
		R	0.78	0.83	0.83	0.75	0.81	0.82	0.94	0.96	0.51	0.73
		F	0.81	0.84	0.85	0.78	0.85	0.84	0.92	0.97	0.60	0.80
AutoNER	Dictionary + Embeddings	P	0.71	0.77	0.91	0.83	0.74	0.65	0.55	**0.68**	0.35	0.82
		R	0.47	0.57	0.55	0.61	0.20	0.59	0.65	0.50	0.37	0.28
		F	0.57	0.66	0.68	0.70	0.32	0.62	0.60	0.58	0.36	0.42
LMEx	Dictionary	P	**0.79**	**0.82**	**0.93**	**0.88**	0.89	0.67	0.54	0.65	0.49	**0.84**
		R	0.51	0.59	0.58	0.64	0.23	0.61	0.76	0.65	0.37	0.29
		F	0.62	**0.69**	**0.71**	**0.74**	0.37	0.63	0.63	**0.65**	0.42	0.43
KnowEx	Dictionary + Embeddings + SME-in-the-loop	P	0.77	0.60	0.59	0.47	**1.00**	**0.71**	**0.97**	0.53	**0.57**	0.78
		R	**0.66**	**0.64**	**0.85**	**0.71**	**0.34**	**0.73**	**0.84**	**0.81**	**0.59**	**0.69**
		F	**0.71**	0.62	0.70	0.56	**0.51**	**0.72**	**0.90**	0.64	**0.58**	**0.73**

leading to an increase in F1 of 0.12 (a percentage increase of around 22%). This performance reflects an improvement in precision on five datasets out of the 10, an improvement of 10 out of 10 for recall, and an improvement of 8 out of 10 in terms of F1. Using the Mann-Whitney U test, with an alpha level of $P < 0.05$, our results achieved a statistically significant $P = 0.036$. Relative to our baseline dictionary-based NER approach, which suffered from low recall, our full pipeline improved recall across the board on all datasets. Specifically, it showed an improvement on 6 out of 10 datasets in terms of F1 scores (0.17 on average), and for the remaining four corpora, it had an average decrease in F1 of only around 6.8% due to the decrease in precision.

The technique, when compared with its supervised counterpart (Flair) [3], provides a much cheaper alternative though sacrificing 0.16 F1 points, on average. However, a reasonable increase in SMEs effort can improve the performance of the pipeline further. Additionally, a more precise user study to calculate the cost of each method is a natural next step. However, as in Section 1.1.3, supervised techniques are still much more expensive

computationally and in terms of time and effort than our Green AI solution.

4.6 Related Work

The KnowEx technique relates to work in the areas of entity set expansion, knowledge-enhanced NER, and dictionary, weakly-supervised, and regex-based entity extraction.

Dictionary-based NER techniques such as AutoNER [118] extract noisy candidate phrases using keyphrase extraction techniques. AutoNER then learns a neural network-based sequence labeling model that tries to minimize the noise effect of distant supervision using their tagging scheme, Tie or Break, which helps in predicting whether two adjacent tokens are part of the same entity or not. However, the technique is still prone to noise and lack of coverage of off-the-shelf dictionaries [2], and does not scale well to multiple domains (see Section 4.5.2). Similarly, our dictionary-based technique in Chapter 2 and in [5] used a language model in addition to a set of domain-specific stop word lists. The use of the stop word list was able to partially address the problem of incorrect extractions, which improved precision but harmed recall. KnowEx improves these by learning a feature-rich classification model with minimal supervision that still uses internal evidence but also incorporates external evidence that (1) induces regular expressions to improve recall, (2) eliminates the need for stopword lists which improves scalability, and (3) filters the extractions of the pipeline further, which improves specificity.

The dictionary learning NER approach by [94] starts by creating a high-recall/low-precision candidates set and then learns a classifier to filter out the set and update the dictionary. However, they use handcrafted domain-specific rules to extract candidate entities, which we induce automatically using regular expressions. On the other hand, the IKE tool [23], an interactive pattern-based extraction method, similar to ours, has a SME-in-the-loop to provide feedback on the pipeline extractions. However, it requires SME inputs in multiple iterations while ours requires only one iteration. The idea of candidate entity

extraction and ranking using diverse features is also similar to the work on entity set expansion (ESE) in Chapter 3 and in [126, 120, 4]. ESE requires a seed set to allow the extraction and retrieval of similar entities based on their features set similarity. Instead, we use LMEx extractions and then rank-order the extractions based on their features in a similar manner to allow for minimal supervision.

Other techniques, such as [2], use only off-the-shelf dictionaries. The technique slightly underperforms state-of-the-art fully-supervised deep neural network word plus character models [19] with only 0.26% F1-Score difference on the English datasets. For the Drug Name Recognition (DNR) task, the state-of-the-art technique [73], which is based on word embeddings plus semantic features based on drug dictionaries, beat the state-of-the-art Deep Neural Network-based techniques on the same task [136]. While the deep learning method does not require dictionaries, it is still a supervised method that requires fully labeled datasets and is more computationally expensive (a cost we seek to minimize).

The use of regular expressions to extract entities is advantageous, given a low budget. However, given a sizable budget, data labeling and supervised approaches are known to perform better than rule-based methods [139]. Many techniques infer regular expressions for entity extraction [121, 9, 140, 13]. However, the majority of these focus on learning regular expressions that extract entities with a uniform structure, such as phone numbers, course numbers, and dates. In contrast, our approach expands the set of candidate entities by learning from the POS patterns synthesized from the extractions of the dictionary-based method.

Recent work used knowledge to improve the performance of NER. Seyler et al. [116] used background knowledge to expand the set of features for CRF, which ultimately improved the performance but found to have a trade-off with more external knowledge. Other neural-based NER methods [28, 74] also improved the performance of their models using dictionaries, showing the value of using external knowledge relative to pure data-driven deep neural net approaches.

4.7 Conclusions, Limitations, and Future Steps

In this chapter, I presented our adaptable, minimally-supervised entity extraction method that is using domain-specific knowledge (i.e., dictionaries), general-purpose knowledge (e.g., Wikipedia categories and contextual word embeddings), and rich lexical and syntactic features that showed improvement of up to 22% in average F1 score over other multi-domain dictionary-based NER methods. The approach *generalizes well* to multiple domains, enhancing recall while maintaining precision in the face of noise. It requires only off-the-shelf dictionaries, online knowledge, and minimal supervision in comparison with other methods.

We did not test our method on social media corpora, but we expect a diverse set of features to compensate for the noise caused by such ungrammatical text. Additionally, we emulated our SMEs-in-the-loop using gold data. In the future, conducting a user study is a possibility to account for noisy user input and measure the time savings while using our minimally supervised pipeline. Finally, we see the strong potential of the approach when integrated with an active-learning-enabled pipeline.

Conclusions and Future Directions

In this dissertation, I presented our knowledge-enabled named entity recognition (NER) methods as effective alternatives to the techniques that require accurate and large amounts of gold annotations and challenged by the evolving content in text streams. Presently, the availability of knowledge and pre-trained machine learning models online can lower the cost of NER, and incorporating them in our pipelines is straightforward. Additionally, requiring minimal supervision or feedback from subject matter experts (SMEs) makes their involvement as human-in-the-loop a possibility rather than outsourcing the labels to workers in services such as Amazon's Mechanical Turk. I proposed three NER techniques that require minimal/reduced levels of supervision via effectively using background knowledge and exploiting the models' certainty and knowledge to lower the cost and improve the accuracy of extraction.

Although supervised and rules-based NER techniques have proven useful for generic or domain-specific use cases, they still suffer from the high computational cost, sparsity of labels or rules (therefore incurring a high cost), infeasibility in the face of challenging use cases (such as for stream processing), and inability to conform to user-centered requirements (such as ease of data annotation). Our techniques, on the other hand, lower the cost of NER, satisfy practical and user-centered requirements, and provide scalable and adaptable NER solutions.

In Chapter 2, I presented our dictionary-based location extraction solution from text streams. I showed how augmenting and filtering dictionaries is essential to improving the

accuracy of NER (the same behavior can be seen in Chapter 4). This includes how the use of dictionaries allowed us to support context-aware computing by extracting location names from streams of text, in an online fashion, even when challenged by ill-formed, nonstandard, and ungrammatical social media text. The technique outperformed ten other location name NER techniques on our evaluation set by at least 33% F1 improvement.

In Chapter 3, I presented our sparse entity extraction technique. The main focus of that technique was to tackle practical user-centered requirements such as privacy preservation by allowing SMEs to label data instead of costly crowdsourcing the labeling. Hence, the need for cost reduction (in terms of the number of labels, the labeling technique, and the complexity of the labeling task). Eventually, the technique allowed us to label for a single entity class (to reduce complexity) and cut (on average) around 45% of the data with the help of a hybrid approach that combined entity set expansion, active learning, and a human-in-the-loop. This makes the NER technique attractive and practical for SMEs in the industrial setting (where less time and cost, and high accuracy is desirable).

In the last NER technique in Chapter 4, I presented our knowledge-enabled NER technique. The main focus of that technique was to overcome the challenges to the domain-agnostic NER technique from Chapter 2, including dictionary incompleteness and scalability to multiple domains. At the end, the use of rich multi-view features (for representation enhancement of candidate entities) and a SME-in-the-loop allowed us to scale the NER solution to multiple domains and to comfortably improve recall with a modest sacrifice of precision while using off-the-shelf dictionaries (not requiring careful cleaning, edits, and adaptation to each entity class).

There are multiple future directions to build on top of the work in this dissertation, some stemming from the limitations of the presented approaches:

- **Out of dictionary entity detection:** In Chapter 4, I presented our KnowEx approach, which partially solved the limitation imposed on the language model extractor in Chapter 2 due to dictionaries incompleteness. The KnowEx approach automatically

induces regular expressions and learns to classify noisy candidate entities with the help of a SME-in-the-loop. While that partially solved the problem, the technique still suffers from the class imbalance between the valid and invalid candidates. Therefore, there is a need to find alternatives to the currently used classifier to work on top of the high-precision/low-recall dictionary extraction.

- **Improving the minimal supervision:** The context-agnostic labeling criterion we implemented in this dissertation queries SMEs to provide binary valid/invalid labels to minimize the labeling cost. However, this labeling technique introduced some noise, as I presented in Chapter 4. Hence, there is a need for developing a technique that can score the labels for individual instances and only accept the binary labels if the score is above a certain threshold, for example. Alternately, using a keyphrase extraction technique might help in ruling out partial matches (due to the compositionality of mentions) to not label them erroneously (e.g., labeling "cancer" as a disease name rather than the full mention "colon cancer").

- **Detailed user study:** In the techniques I presented in this dissertation, we emulated SMEs-in-the-loop using gold data. Conducting a more detailed user study while having SMEs-in-the-loop would, therefore, have the following benefits: (1) provide an accurate measure of how much cost reduction our techniques were able to provide, (2) provide a more detailed view on how prone our techniques are when faced with noisy SME labels, and (3) provide a critical view on the techniques when having multiple SMEs and how to account and deal with their disagreements in their labels.

Gathering explicit and contextual features for entity recognition is challenging due to data sparsity, the need for reliable annotation, and timeliness in the face of evolving data streams. Background knowledge, minimal supervision from subject-matter-experts, and machine learning models' certainty can be leveraged to develop reliable named entity recognizers at a reduced cost. As a conclusion and an answer to this thesis statement, the

techniques presented in this dissertation, when compared with state-of-the-art supervised and rule-based NER systems, require fewer labels, exploit readily available knowledge, scale to multiple domains, honor practical and user-centered requirements. Hence, it compensates for the need to rigorously supervise and prepare vast amounts of labeled data via minimal supervision and a kind of distant supervision. Ultimately, in our generalizable techniques, we tried to solve pressing technical and practical problems that I hope would make NER more attractive to use. Finally, I hope that my research will motivate other researchers in the future to pursue practical aspirations and motivate them to open source their solutions, which would accelerate the advancement of our field.

Bibliography

[1] Asma Ben Abacha and Pierre Zweigenbaum. Medical entity recognition: A comparison of semantic and statistical methods. In *Proceedings of BioNLP 2011 Workshop*, pages 56–64. Association for Computational Linguistics, 2011.

[2] Rodrigo Agerri and German Rigau. Robust multilingual named entity recognition with shallow semi-supervised features. *Artif. Intell.*, 238(C):63–82, September 2016.

[3] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, 2018.

[4] Hussein Al-Olimat, Steven Gustafson, Jason Mackay, Krishnaprasad Thirunarayan, and Amit Sheth. A Practical Incremental Learning Framework For Sparse Entity Extraction. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 700–710, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[5] Hussein Al-Olimat, Krishnaprasad Thirunarayan, Valerie Shalin, and Amit Sheth. Location Name Extraction from Targeted Text Streams using Gazetteer-based Statistical Language Models. In *Proceedings of the 27th International Conference on*

Computational Linguistics, pages 1986–1997, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[6] Hussein S. Al-Olimat, Valerie L. Shalin, Krishnaprasad Thirunarayan, and Joy Prakash Sain. Towards geocoding spatial expressions (vision paper). In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '19, page 75–78, New York, NY, USA, 2019. Association for Computing Machinery.

[7] Stephen H Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alexander Ratner, Braden Hancock, Houman Alborzi, et al. Snorkel drybell: A case study in deploying weak supervision at industrial scale. *arXiv preprint arXiv:1812.00417*, 2018.

[8] Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. How noisy social media text, how diffrnt social media sources? In *International Joint Conference on Natural Language Processing*, pages 356–364, 2013.

[9] Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao. Active learning of regular expressions for entity extraction. *IEEE transactions on cybernetics*, 48(3):1067–1080, 2018.

[10] Oliver Bender, Franz Josef Och, and Hermann Ney. Maximum entropy models for named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 148–151, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[11] Alexandre Bonnasseau and Open Knowledge Foundation OKF. Major cities of the world (CSV File). Accessed: 2019-05-06.

[12] Kalina Bontcheva, Leon Derczynski, Adam Funk, Mark A Greenwood, Diana Maynard, and Niraj Aswani. Twitie: An open-source information extraction pipeline for microblog text. In *RANLP*, pages 83–90, 2013.

[13] Falk Brauer, Robert Rieger, Adrian Mocan, and Wojciech M Barczynski. Enabling information extraction by inference of regular expressions from sample entities. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1285–1294. ACM, 2011.

[14] Alexander Budanitsky and Graeme Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47, 2006.

[15] John M Carroll. Nameheads. *Cognitive science*, 7(2):121–153, 1983.

[16] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[17] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

[18] Laura Chiticariu, Yunyao Li, and Frederick R Reiss. Rule-based information extraction is dead! long live rule-based information extraction systems! In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 827–832, 2013.

[19] Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*, 2015.

[20] Noam Chomsky and David W Lightfoot. *Syntactic structures*. Walter de Gruyter, 2002.

[21] Harry Collins. *Tacit and explicit knowledge*. University of Chicago Press, 2010.

[22] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. Gate: an architecture for development of robust hlt applications. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 168–175. Association for Computational Linguistics, 2002.

[23] Bhavana Dalvi, Sumithra Bhakthavatsalam, Chris Clark, Peter Clark, Oren Etzioni, Anthony Fader, and Dirk Groeneveld. IKE - an interactive tool for knowledge extraction. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 12–17, San Diego, CA, June 2016. Association for Computational Linguistics.

[24] Nilesh Dalvi, Marian Olteanu, Manish Raghavan, and Philip Bohannon. Deduplicating a places database. In *Proceedings of the 23rd international conference on World wide web*, pages 409–418. Association for Computing Machinery (ACM), 2014.

[25] Allan Peter Davis, Thomas C Wiegers, Phoebe M Roberts, Benjamin L King, Jean M Lay, Kelley Lennon-Hopkins, Daniela Sciaky, Robin Johnson, Heather Keating, Nigel Greene, et al. A ctd–pfizer collaboration: manual curation of 88 000 scientific articles text mined for drug–disease and drug–phenotype interactions. *Database*, 2013, 2013.

[26] Leon Derczynski, Diana Maynard, Giuseppe Rizzo, Marieke van Erp, Genevieve Gorrell, Raphaël Troncy, Johann Petrak, and Kalina Bontcheva. Analysis of named entity recognition and linking for tweets. *Information Processing & Management*, 51(2):32–49, 2015.

[27] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[28] Ruixue Ding, Pengjun Xie, Xiaoyan Zhang, Wei Lu, Linlin Li, and Luo Si. A neural multi-digraph model for Chinese NER with gazetteers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1462–1467, Florence, Italy, July 2019. Association for Computational Linguistics.

[29] Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10, 2014.

[30] Doug Downey, Matthew Broadhead, and Oren Etzioni. Locating complex named entities in web text. In *IJCAI*, volume 7, pages 2733–2739, 2007.

[31] Mark Dredze, Michael J Paul, Shane Bergsma, and Hieu Tran. Carmen: A twitter geolocation system with applications to public health. In *AAAI workshop on expanding the boundaries of health informatics using AI (HIAI)*, pages 20–24. Citeseer, 2013.

[32] Alexander Hussam Elkholy, Balasubramanian Kandaswamy, Steven Matt Gustafson, and Hussein S Al-Olimat. Machine assisted learning of entities, March 26 2019. US Patent App. 10/242,320.

[33] Larry Smith et. al. Overview of biocreative ii gene mention recognition. *Genome Biology*, 9(2):S2, Sep 2008.

[34] Speed Reading International Execuread. Speed Reading Facts. Accessed: 2019-12-06.

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

[36] Zalando Research Flair. A very simple framework for state-of-the-art NLP. Accessed: 2019-12-06.

[37] Jason Fries, Sen Wu, Alex Ratner, and Christopher Ré. Swellshark: A generative model for biomedical named entity recognition without labeled data. *arXiv preprint arXiv:1704.06360*, 2017.

[38] Judith Gelernter and Shilpa Balaji. An algorithm for local geoparsing of microtext. *GeoInformatica*, 17(4):635–667, 2013.

[39] Judith Gelernter, Gautam Ganesh, Hamsini Krishnakumar, and Wei Zhang. Automatic gazetteer enrichment with user-geocoded data. In *Proceedings of the Second ACM SIGSPATIAL International Workshop on Crowdsourced and Volunteered Geographic Information*, pages 87–94. Association for Computing Machinery (ACM), 2013.

[40] Judith Gelernter and Wei Zhang. Cross-lingual geo-parsing for non-structured data. In *Proceedings of the 7th Workshop on Geographic Information Retrieval*, pages 64–71. Association for Computing Machinery (ACM), 2013.

[41] David Gertman, Harold Blackman, Julie Marble, James Byers, Curtis Smith, et al. The spar-h human reliability analysis method. *US Nuclear Regulatory Commission*, 230, 2005. Page 35.

[42] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[43] Ralph Grishman and Yifan He. An information extraction customizer. In Petr Sojka, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, *Text, Speech and Dialogue*, pages 3–10, Cham, 2014. Springer International Publishing.

[44] Sonal Gupta. *Distantly Supervised Information Extraction Using Bootstrapped Patterns*. PhD thesis, Stanford University, 2015.

[45] Sonal Gupta and Christopher D Manning. Improved pattern learning for bootstrapped entity extraction. In *CoNLL*, pages 98–108, 2014.

[46] David Guthrie, Ben Allison, Wei Liu, Louise Guthrie, and Yorick Wilks. A closer look at skip-gram modelling. In *Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC-2006)*, pages 1–4, 2006.

[47] L Melissa Guzman. USA Fast Food Restaurants (CSV File). Accessed: 2019-05-06.

[48] Bo Han, Antonio Jimeno Yepes, Andrew MacKinlay, and Qiang Chen. Identifying twitter location mentions. In *Australasian Language Technology Association Workshop 2014*, page 157, 2014.

[49] Qiang Hao, Rui Cai, Yanwei Pang, and Lei Zhang. From one tree to a forest: a unified solution for structured web data extraction. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 775–784. ACM, 2011.

[50] Mike Hazas, James Scott, and John Krumm. Location-aware computing comes of age. *Computer*, 37(2):95–97, 2004.

[51] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328. IEEE, 2008.

[52] Thi Bich Ngoc Hoang and Josiane Mothe. Location extraction from tweets. *Information Processing & Management*, 54(2):129–144, 2018.

[53] Zongcheng Ji, Aixin Sun, Gao Cong, and Jialong Han. Joint recognition and linking of fine-grained locations from tweets. In *Proceedings of the 25th International Conference on World Wide Web*, pages 1271–1281. International World Wide Web Conferences Steering Committee, 2016.

[54] Shruti Kar, Hussein S. Al-Olimat, Krishnaprasad Thirunarayan, Valerie Shalin, Amit Sheth, and Srinivasan Parthasarathy. D-record: Disaster response and relief coordination pipeline. In *Proceedings of the ACM SIGSPATIAL International Workshop on Advances in Resilient and Intelligent Cities (ARIC 2018)*. Association for Computing Machinery, 2018.

[55] Max Kaufmann and Jugal Kalita. Syntactic normalization of twitter messages. In *International conference on natural language processing, Kharagpur, India*, 2010.

[56] Panayiota Kendeou and Paul Van Den Broek. The effects of prior knowledge and text structure on comprehension processes during reading of scientific texts. *Memory & cognition*, 35(7):1567–1577, 2007.

[57] Arbaz Khan, Maria Vasardani, and Stephan Winter. Extracting spatial information from place descriptions. In *Proceedings of The First ACM SIGSPATIAL International Workshop on Computational Models of Place*, COMP '13, pages 62:62–62:69, New York, NY, USA, 2013. ACM.

[58] Mahnoosh Kholghi, Laurianne Sitbon, Guido Zuccon, and Anthony Nguyen. Active learning reduces annotation time for clinical concept extraction. *International journal of medical informatics*, 106:25–31, 2017.

[59] Kim. Kim's Body Breakdown. Accessed: 2019-05-06.

[60] Seokhwan Kim, Yu Song, Kyungduk Kim, Jeong-Won Cha, and Gary Geunbae Lee. Mmr-based active machine learning for bio named entity recognition. In *Proceed-*

ings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers, pages 69–72. Association for Computational Linguistics, 2006.

[61] Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick Reiss, Shivakumar Vaithyanathan, and Huaiyu Zhu. Systemt: a system for declarative information extraction. *ACM SIGMOD Record*, 37(4):7–13, 2009.

[62] Beth Levin. *English verb classes and alternations: A preliminary investigation*. University of Chicago press, 1993.

[63] Chenliang Li and Aixin Sun. Fine-grained location extraction from tweets with temporal awareness. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 43–52, New York, NY, USA, 2014. ACM.

[64] Chenliang Li, Jianshu Weng, Qi He, Yuxia Yao, Anwitaman Datta, Aixin Sun, and Bu-Sung Lee. Twiner: named entity recognition in targeted twitter stream. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 721–730. Association for Computing Machinery (ACM), 2012.

[65] Guoliang Li, Jun Hu, Jianhua Feng, and Kian-lee Tan. Effective location identification from microblogs. In *Data Engineering (ICDE), 2014 The Institute of Electrical and Electronics Engineers (IEEE) 30th International Conference on*, pages 880–891. The Institute of Electrical and Electronics Engineers (IEEE), 2014.

[66] Yehoshua Zvi Licht, David Allen Turner, and Joseph Arnold White. Location context aware computing, November 30 2017. US Patent App. 15/166,740.

[67] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

[68] John Lingad, Sarvnaz Karimi, and Jie Yin. Location extraction from disaster-related microblogs. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1017–1020. Association for Computing Machinery (ACM), 2013.

[69] Todd Lingren, Louise Deleger, Katalin Molnar, Haijun Zhai, Jareen Meinzen-Derr, Megan Kaiser, Laura Stoutenborough, Qi Li, and Imre Solti. Evaluating the impact of pre-annotation on annotation speed and potential bias: natural language processing gold standard development for clinical named entity recognition in clinical trial announcements. *Journal of the American Medical Informatics Association*, 21(3):406–413, 2013.

[70] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1):503–528, Aug 1989.

[71] Fei Liu, Maria Vasardani, and Timothy Baldwin. Automatic identification of locative expressions from social media text: A comparative analysis. In *Proceedings of the 4th International Workshop on Location and the Web*, pages 9–16. Association for Computing Machinery (ACM), 2014.

[72] Jingjing Liu, Panupong Pasupat, Scott Cyphers, and Jim Glass. Asgard: A portable architecture for multilingual dialogue systems. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8386–8390. IEEE, 2013.

[73] Shengyu Liu, Buzhou Tang, Qingcai Chen, and Xiaolong Wang. Effects of semantic features on machine learning-based drug name recognition systems: word embeddings vs. manually constructed dictionaries. *Information*, 6(4):848–865, 2015.

[74] Tianyu Liu, Jin-Ge Yao, and Chin-Yew Lin. Towards improving neural named entity recognition with gazetteers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5301–5307, Florence, Italy, July 2019. Association for Computational Linguistics.

[75] Shervin Malmasi and Mark Dras. Location mention detection in tweets and microblogs. In *International Conference of the Pacific Association for Computational Linguistics (PACL)*, pages 123–134. Springer, 2015.

[76] Pranav Maneriker, Nikhita Vedula, Hussein S. Al-Olimat, Jiayong Liang, Omar El-Khoury, Ethan Kubatko, Desheng Liu, Krishnaprasad Thirunarayan, Valerie Shalin, Amit Sheth, and Srinivasan Parthasarathy. A pipeline for disaster response and relief coordination. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 1337–1340, New York, NY, USA, 2019. ACM.

[77] Inderjeet Mani and I Zhang. knn approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets*, volume 126, 2003.

[78] Gideon S Mann and Andrew McCallum. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proceedings of ACL-08: HLT*, pages 870–878, 2008.

[79] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103, 2010.

[80] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999.

[81] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

[82] Lynne M Markus. Toward a theory of knowledge reuse: Types of knowledge reuse situations and factors in reuse success. *Journal of management information systems*, 18(1):57–93, 2001.

[83] Koji Matsuda, Akira Sasaki, Naoaki Okazaki, and Kentaro Inui. Annotating geographical entities on microblog text. In *The 9th Linguistic Annotation Workshop held in conjuncion with North American Chapter of the Association for Computational Linguistics (NAACL) 2015*, page 85, 2015.

[84] David McDonald. Internal and external evidence in the identification and semantic categorization of proper names. *Acquisition of Lexical Knowledge from Text*, 1993.

[85] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8. ACM, 2011.

[86] Stuart E Middleton, Lee Middleton, and Stefano Modafferi. Real-time crisis mapping of natural disasters using social media. *The Institute of Electrical and Electronics Engineers (IEEE) Intelligent Systems*, 29(2):9–17, 2014.

[87] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[88] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[89] Nicholas Ross Milton. *Knowledge acquisition in practice: a step-by-step guide*. Springer Science & Business Media, 2007.

[90] Bonan Min and Ralph Grishman. Fine-grained entity set refinement with user feedback. *Information Extraction and Knowledge Acquisition*, page 2, 2011.

[91] Robert Munro. Subword and spatiotemporal models for identifying actionable information in haitian kreyol. In *Proceedings of the fifteenth conference on computational*

natural language learning, pages 68–77. Association for Computational Linguistics (ACL), 2011.

[92] Roberto Navigli and Simone Paolo Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217 – 250, 2012.

[93] Claire Nédellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo, and Pierre Zweigenbaum. Overview of BioNLP shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 1–7, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[94] Arvind Neelakantan and Michael Collins. Learning dictionaries for named entity recognition using minimal supervision. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 452–461, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

[95] Chris Nikolopoulos. *Expert systems: introduction to first and second generation and hybrid knowledge based systems*. Marcel Dekker, Inc., 1997.

[96] Sebastian Ruder NLP-progress. NLP-progress (Named entity recognition). Accessed: 2019-12-06.

[97] Peter Norvig. Natural language corpus data. In T. Segaran and J. Hammerbacher, editors, *Beautiful Data*, chapter 14, pages 219–242. O'Reilly Media, 2009.

[98] Brendan O'Connor, Michel Krieger, and David Ahn. Tweetmotif: Exploratory search and topic summarization for twitter. In *ICWSM*, pages 384–385, 2010.

[99] Tomoko Ohta, Yuka Tateisi, Jin-Dong Kim, Sang-Zoo Lee, and Jun'ichi Tsujii. Genia corpus: A semantically annotated corpus in molecular biology domain. In *Pro-*

ceedings of the ninth International Conference on Intelligent Systems for Molecular Biology (ISMB 2001) poster session, volume 68, 2001.

[100] Lluís Padró and Evgeny Stanilovsky. FreeLing 3.0: Towards Wider Multilinguality. In *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*, Istanbul, Turkey, May 2012. ELRA.

[101] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[102] Steven Pinker. *The language instinct: How the mind creates language*. Penguin UK, 2003.

[103] Jakub Piskorski and Maud Ehrmann. On named entity recognition in targeted twitter streams in polish. In *The 4th Biennial International Workshop on Balto-Slavic Natural Language Processing: ACL*, pages 84–93. Citeseer, 2013.

[104] Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Haris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval-2014)*, pages 27–35, 2014.

[105] Sampo Pyysalo, Filip Ginter, Hans Moen, Tapio Salakoski, and Sophia Ananiadou. Distributional semantic resources for biomedical text mining. In *Proceedings of the 5th International Symposium on Languages in Biology and Medicine*, pages 39–44, 2013.

[106] Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. pigeo: A python geotagging tool. *Proceedings of ACL-2016 System Demonstrations*, pages 127–132, 2016.

[107] Ines Rehbein, Josef Ruppenhofer, and Caroline Sporleder. Assessing the benefits of partial automatic pre-labeling for frame-semantic annotation. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 19–26. Association for Computational Linguistics, 2009.

[108] L.B. Resnick, J.M. Levine, and S.D. Teasley. *Perspectives on Socially Shared Cognition*. American Psychological Association, 1991.

[109] Alan Ritter, Sam Clark, and Oren Etzioni. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics (ACL), 2011.

[110] Xin Rong, Zhe Chen, Qiaozhu Mei, and Eytan Adar. Egoset: Exploiting word ego-networks and user-generated ontology for multifaceted set expansion. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 645–654. ACM, 2016.

[111] Kugatsu Sadamitsu, Kuniko Saito, Kenji Imamura, and Yoshihiro Matsuo. Entity set expansion using interactive topic information. In *PACLIC*, pages 108–116, 2012.

[112] Luis Sarmento, Valentin Jijkuon, Maarten de Rijke, and Eugenio Oliveira. More like these: growing entity classes from seeds. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 959–962. ACM, 2007.

[113] Axel Schulz, Aristotelis Hadjakos, Heiko Paulheim, Johannes Nachtwey, and Max Mühlhäuser. A multi-indicator approach for geolocalization of tweets. In *ICWSM*, 2013.

[114] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green ai. *arXiv preprint arXiv:1907.10597*, 2019.

[115] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics, 2008.

[116] Dominic Seyler, Tatiana Dembelova, Luciano Del Corro, Johannes Hoffart, and Gerhard Weikum. A study of the importance of external knowledge in the named entity recognition task. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 241–246, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[117] J. Shang, J. Liu, M. Jiang, X. Ren, C. R. Voss, and J. Han. Automated phrase mining from massive text corpora. *IEEE Transactions on Knowledge and Data Engineering*, 30(10):1825–1837, Oct 2018.

[118] Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. Learning named entity tagger using domain-specific dictionary. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2054–2064, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[119] Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[120] Jiaming Shen, Zeqiu Wu, Dongming Lei, Jingbo Shang, Xiang Ren, and Jiawei Han. Setexpan: Corpus-based set expansion via context feature selection and rank ensemble. In Michelangelo Ceci, Jaakko Hollmén, Ljupčo Todorovski, Celine Vens, and Sašo Džeroski, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 288–304, Cham, 2017. Springer International Publishing.

[121] Stanley Simoes, P Deepak, Munu Sairamesh, Deepak Khemani, and Sameep Mehta. Content and context: Two-pronged bootstrapped learning for regex-formatted entity extraction. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[122] Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. BRAT: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations Session at EACL 2012*, Avignon, France, April 2012. Association for Computational Linguistics (ACL).

[123] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy, July 2019. Association for Computational Linguistics.

[124] Tony Stubblebine. *Regular Expression Pocket Reference: Regular Expressions for Perl, Ruby, PHP, Python, C, Java and. NET*. " O'Reilly Media, Inc.", 2007.

[125] Evan A Sultanik and Clayton Fink. Rapid geotagging and disambiguation of social media text via an indexed gazetteer. *Proceedings of International conference on Information Systems for Crisis Response and Management (ISCRAM)*, 12:1–10, 2012.

[126] Fangbo Tao, Bo Zhao, Ariel Fuxman, Yang Li, and Jiawei Han. Leveraging pattern semantics for extracting entities in enterprises. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1078–1088. International World Wide Web Conferences Steering Committee, 2015.

[127] Erik F Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics, 2003.

[128] Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. Accelerating the annotation of sparse named entities by dynamic sentence selection. *BMC bioinformatics*, 9(11):S8, 2008.

[129] Stephen P Turner. *Understanding the tacit*. Routledge, 2014.

[130] Wikipedia WClarke. 2016 Louisiana floods map of parishes declared federal disaster areas.png. Accessed: 2019-05-06.

[131] Chih-Hsuan Wei, Yifan Peng, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Jiao Li, Thomas C Wiegers, and Zhiyong Lu. Overview of the biocreative v chemical disease relation (cdr) task. In *Proceedings of the fifth BioCreative challenge evaluation workshop*, pages 154–166, 2015.

[132] Davy Weissenbacher, Tasnia Tahsin, Rachel Beard, Mari Figaro, Robert Rivera, Matthew Scotch, and Graciela Gonzalez. Knowledge-driven geospatial location resolution for phylogeographic models of virus migration. *Bioinformatics*, 31(12):i348–i356, 2015.

[133] Wikipedia.org. List of organisms by chromosome count. Accessed: 2019-05-06.

[134] Patrick H Winston. Learning structural descriptions from examples. 1970.

[135] Vikas Yadav and Steven Bethard. A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158. Association for Computational Linguistics, 2018.

[136] Vikas Yadav, Rebecca Sharp, and Steven Bethard. Deep affix features improve neural named entity recognizers. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 167–172, 2018.

[137] Nan Ye, Wee S Lee, Hai L Chieu, and Dan Wu. Conditional random fields with high-order features for sequence labeling. In *Advances in Neural Information Processing Systems*, pages 2196–2204, 2009.

[138] Jie Yin, Sarvnaz Karimi, and John Lingad. Pinpointing locational focus in microblogs. In *Proceedings of the 2014 Australasian Document Computing Symposium*, page 66. Association for Computing Machinery (ACM), 2014.

[139] Shanshan Zhang, Lihong He, Eduard Dragut, and Slobodan Vucetic. How to invest my time: Lessons from human-in-the-loop entity extraction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 2305–2313, New York, NY, USA, 2019. ACM.

[140] Shanshan Zhang, Lihong He, Slobodan Vucetic, and Eduard Dragut. Regular expression guided entity mention mining from noisy web data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1991–2000, 2018.