U-AIDA: a Customizable System for Named Entity Recognition, Classification, and Disambiguation

Mohamed Amir Yosef

Dissertation
zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

Saarbrücken
2015

Dean	Prof. Dr. Markus Bläser
Colloqium	11.12.2015 Saarbrücken

Examination Board

Supervisor and Reviewer	Prof. Dr.-Ing. Gerhard Weikum
Reviewer	Prof. Dr. Klaus Berberich
Reviewer	Prof. Dr. Marc Spaniol
Chairman	Prof. Dr. Dietrich Klakow
Research Assistant	Dr. Johannes Hoffart

Abstract

Recognizing and disambiguating entities such as people, organizations, events or places in natural language text are essential steps for many linguistic tasks such as information extraction and text categorization. A variety of named entity disambiguation methods have been proposed, but most of them focus on Wikipedia as a sole knowledge resource. This focus does not fit all application scenarios, and customization to the respective application domain is crucial.

This dissertation addresses the problem of building an easily customizable system for named entity disambiguation. The first contribution is the development of a universal and flexible architecture that supports plugging in different knowledge resources. The second contribution is utilizing the flexible architecture to develop two domain-specific disambiguation systems. The third contribution is the design of a complete pipeline for building disambiguation systems for languages other than English that have poor annotated resources such as Arabic. The fourth contribution is a novel approach that performs fine-grained type classification of names in natural language text.

Kurzfassung

Das Erkennen und die Disambiguierung von Entitäten wie etwa Personen, Organisationen oder Orte in natürlichsprachigem Text sind wertvolle Hilfsmittel für zahlreiche linguistische Aufgaben Biespielanwendungen sind Informationsextraktion oder die Kategorisierung von Texten. In diesem Kontext sind eine Vielzahl von Verfahren zur Disambiguierung erforscht worden. Allerdings basieren die meisten dieser Verfahren lediglich auf dem aus Wikipedia extrahierbaren "Wissen". Diese Fokussierung eignet sich jedoch keineswegs für alle Anwendungsszenarien, weshalb eine Anpassung an die jeweils vorliegende Anwendungsdomäne besonders wichtig ist. Diese Dissertation befasst sich daher mit dem Entwurf eines Universell einsetzbaren und individuell konfigurierbaren Systems zur Disambiguierung von Entitätsnamen.

Der erste Beitrag dieser Arbeit ist die Entwicklung einer universell einsatzfähigen und anpassbaren Architektur, die das Einbinden unterschiedlicher Wissensquellen ermöglicht. Darauf aufbauend wird die Flexibilität der vorgestellten Architektur mittels zweier domänen-spezifischer Anwendungen belegt. Darüber hinaus wird die Vielseitikeit des Verfahrens durch den Entwurf eines kompletten Verarbeitungsprozess für resourcenarme Sprachen am Beispiel der arabischen Sprache gezeigt. Abschließend wird ein neuartiger Ansatz zur feingranularen Typisierung von benannten Entitäten in natürlichsprachigem Text vorgestellt.

Summary

Discovering mentions of named entities such as people, events, location or organizations and linking them to canonical entities registered in a knowledge resource is a valuable asset in many linguistic tasks such as semantic search and information extraction. The English Wikipedia is the most widely used knowledge resource in the literature for performing named entity disambiguation. However, the English Wikipedia is only suitable for disambiguating general English text such as English news articles. Developing a disambiguation system for other domains and languages requires major adaptation to fit the specific application scenarios. In addition, the Wikipedia editions for many languages, such as Arabic, are an order of magnitude smaller than the English Wikipedia. Therefore, it is crucial to exploit cross-language evidences to enrich the non-English resources. Finally, some names cannot be disambiguated because they denote entities that do not exist in the underlying knowledge resource. This dissertation makes the following contributions to address the problem of building a universal and customizable disambiguation system.

U-AIDA Architecture: We developed a universal architecture called U-AIDA for building named entity disambiguation solutions. The architecture is flexible and supports plugging in multiple knowledge resources to be used as the underlying repository for named entities. U-AIDA can be easily customized to fit various application scenarios.

Domain-Specific Disambiguation Systems: We leveraged the flexibility of U-AIDA architecture to build two domain-specific systems. The first is developed to handle German documents from the German National Library. It combines a general-purpose knowledge base with a domain-specific knowledge base developed by the German National Library. The second system is geared towards social streams. It considers Twitter as a use case and accordingly adapts different components of U-AIDA.

Disambiguating non-English Text: We designed a complete pipeline for building named entity disambiguation systems capable of processing text of languages with poor annotated resources such as Arabic. We exploited cross-language evidences to enrich these poor resources with the English counterpart. In addition, we incorporated statistical machine translation techniques to translate some of the English resources into the target language. We implemented the system within the U-AIDA framework and tested it on Spanish, Italian and Arabic. Experiments showed up to 8% improvement in precision and recall after applying our data enrichment techniques for the Arabic languages. For Spanish and Italian the improvement was around 4% because of their relatively richer Wikipedias.

Named Entity Classification: Texts from recent news article may contain newly emerging entities that are unknown to the named entity disambiguation system. We developed a machine-learning based approach, called HYENA, to classify names of entities under a fine grained hierarchy of 505 semantic types. We tested our system on different data sets and compared it to state-of-the-art systems. HYENA outperformed other systems on various data sets. In addition, we conducted an extrinsic study on named entity disambiguation to analyze the reduction in search space when applying type-based pruning on the candidate list. Our experiments showed that 17% reduction in search space could be achieved with only 2% drop in precision.

Zusammenfassung

Das Erkennen und die Disambiguierung von Entitäten wie etwa Personen, Organisationen oder Orte in natürlichsprachigem Text sind wertvolle Hilfsmittel für zahlreiche linguistische Aufgaben. Die englische Version der Online-Enzyklopädie Wikipedia ist dabei die am häufigsten verwendete Quelle für die Disambiguierung. Allerdings ist die englischsprachige Wikipedia im wesentlichen "nur" dazu geeignet, englische Nachrichtenartikel zu disambiguieren. Die Entwicklung eines Disambiguierungssystems für andere Szenarien und/oder Sprachen erfordert daher umfassende Anpassungen an das jeweilige Anwendungsgebiet. Zudem ist Wikipedia in vielen anderen Sprachen, wie z.B. dem Arabischen, um (mehrere) Größenordnungen kleiner als die englische Wikipedia. Von daher ist es oftmals erforderlich, inter-linguale Evidenzen zu nutzen, um Wikipedia für weniger verbreitete Sprachen mit den Quellen aus der englischen Wikipedia zu verknüpfen. Schlussendlich gibt es auch noch benannte Entitäten, die überhaupt nicht disambiguiert werden können, weil zu diesen überhaupt kein Eintrag in der Wissensquelle vorhanden ist. Diese Dissertation befasst sich daher mit dem Entwurf eines Universell einsetzbaren und individuell konfigurierbaren Systems zur Disambiguierung von Entitätsnamen.

U-AIDA Architektur: Die universelle U-AIDA Architektur wurde für ein adaptives Disambiguierungssystem dazu entwickelt. Diese Architektur ist flexibel ausgelegt und erlaubt die Einbindung beliebiger Wissensquellen, welche benannte Entitäten enthalten. Zu diesem Zweck kann U-AIDA vielseitig konfiguriert und an nahezu beliebige Anwendungsszenarien angepasst werden.

Domänenspezifische Disambigiguierungssysteme: Die Flexibilität von U-AIDA wurde dazu genutzt, um zwei domänenspezische Systeme zu entwickeln. Das rste System wurde dazu verwendet, deutschsprachige Dokumente der deutschen Nationalbibliothek (DNB) zu bearbeiten. Dazu wird eine allgemeine Wissensbasis mit einer bibliothekarischen Wissenquelle der DNB kombiniert. Das zweite System zielt auf soziale Netzwerke ab. Im konkreten Fall handelt es sich dabei um eine Anpassung von U-AIDA zum Monitoring des Twitter-Nachrichtendiensts.

Disambiguierung nicht englischsprachiger Texte: Um Texte in ressourcenarmen Sprachen wie etwa dem Arabischen zu disambiguieren, haben wir einen vollständigen Verarbeitungsprozess entwickelt. Dazu wurden inter-linguale Evidenzen genutzt, um Wikipedia für weniger verbreitete Sprachen mit den Quellen aus der englischen Wikipedia zu verknüpfen. Zudem wurden statistische Verfahren des maschinellen Lernens dazu eingesetzt, dedizierte

englischsprachige Ressourcen in die ressourcenarme Sprache übersetzt. Zu diesem Zweck wurde das U-AIDA System in Arabisch, Italienisch und Spanisch getestet und evaluiert. Experimente zeigen dabei bis zu 8% Steigerung in Präzision und Ausbeute für Arabisch. Für Italienisch und Spanisch wurde, bedingt durch deren größeren Ausgangsdatenbestand, immerhin noch eine Verbesserung von nahezu 5% erzielt.

Klassifikation benannter Entitäten: Nachrichtenartikel enthalten häufig Entitäten, die aufgrund ihrer erstmaligen Nennung noch nicht in den zugrundeliegenden Wissenbasen registriert sind. Um auch solche Entitäten typisieren zu können, wurde HYENA entwickelt. HYENA basiert auf maschinellem Lernen und ist dazu geeignet, Entitäten in einer feingranularen Hierachie von 505 Typen zu klassifizieren. Das System wurde im Vergleich mit anderen Referenzsystemen evaluiert. Weiterhin wurde HYENA in einer extrinsischen Studie dazu eingesetzt, den Suchraum der Kandidaten bei der Disambiguierung auf die von HYENA vorgegebenen Typen zu reduzieren. Experimente zeigten dass sich bei einer Reduzierung des Suchraums von 17% die Güte der Präzision der Disambiguierung lediglich um 2% reduziert.

To my parents

Acknowledgments

I want to show my deep gratitude to my supervisor Gerhard Weikum for his continuous support throughout my doctoral studies. He always provided me with visionary ideas that made this dissertation possible. In addition, he was understanding of my personal life. I want thank the International Max Planck Research School for Computer Science as well as the Max Planck Society for funding this work and for doing their best to make my research process easier.

I am grateful to Marc Spaniol for his scientific guidance as well as valuable support on the personal level. I want to thank Johannes Hoffart for our collaboration on the AIDA project. He is among the best people I have worked with. I am grateful to other people who contributed to the AIDA project, especially Yusra Ibrahim and Mohamed Gad-Elrab, and to Asia Biega and Fabian Suchanek for their valuable help in developing the multilingual aspect of U-AIDA. I want to thank my colleagues at the Databases and Information Systems group for their friendly environment, most notably, Klaus Berberich, Adam Grycner, Stephan Seufert, and Mohamed Yahya .

Most importantly, I want to thank my parents, Mervat Tawfik and Amir Mansour, and my wife Walaa Ammar. I owe my parents who I am today, they were always super motivating and supporting before and during my PhD. My wife was courageous enough to accept to marry me during my first year in my PhD. She has shared both cheerful and stressful moments with me, and was the best support I got throughout my PhD.

Contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation

Named Entity Recognition and Disambiguation (NERD) is the problem of spotting mentions of named entities such as `PERSONS`, `LOCATIONS` and `ORGANIZATIONS` in natural language text and linking them to canonical entities registered in a Knowledge Base (KB). Consider this example:

> Page is one of the co-founders of Google

NERD identifies "Page" as a mention of a person, and links it to "Larry Page" as the correct entity to which the sentence refers.

The problem of NERD has been widely discussed in the literature. However, research has focused more on producing better disambiguation algorithms and techniques, and less on the customizability of NERD systems to work for different domains as well as different languages. Most of the available NERD systems are tested against English corpora extracted from the news or Wikipedia articles. However, NERD is not a "one-size-fits-all" problem; instead, application-specific solutions are required. In order to design an architecture for a universal NERD framework that can be easily customized to different languages and domains with small effort, different design aspects need to be considered.

For example, unlike English, Arabic is a morphologically rich language that requires more sophisticated processing. Hence, the architecture should allow plugging in different language processing components for different languages.

In addition, most NERD systems are using Wikipedia or a Wikipedia derived KB such as YAGO [25] or Freebase [6] as the underlying KB. While Wikipedia is a compelling choice when disambiguating general news articles, it is by far not adequate for disambiguating text from more specific domains such as health or music. Building a NERD system for different domains requires plugging in one or more KBs other than Wikipedia.

Finally, a universal NERD system should also be able to handle mentions referring to entities that do not exist in the underlying KBs. While they cannot be linked to

any entity in the KB, they could be classified under the type hierarchy of the KB. For example, when processing a sports report about a football match, some players might not be prominent enough to appear in Wikipedia or a similar KB. Nevertheless, the context is rich enough to flag them not only as persons, but as athletes or even football players. Mention classification is of high value for *known* entities as well. Being able to classify a mention as an athlete helps to reduce the search space when applying NERD by filtering out all non-athlete entities from the candidate list. Not only does this potentially improve NERD quality, but it also improves the runtime of NERD methods.

1.2. Terminology

Various technical terms are used throughout the dissertation. Below is a listing of the most important ones.

Named Entity refers to a uniquely identified item in a set of other items that share one or more attributes. For example, "Angela Merkel" is a uniquely identified person in the set of people.

Knowledge Base (KB) is a collection of named entities such as PERSONS, LOCATIONS and ORGANIZATIONS. Each entity is uniquely identified within the KB. There exist semantic relations among entities, and entities are organized under a type hierarchy.

Mention is a surface form that denotes one specific named entity out of a set of potential candidate named entities.

Named Entity Recognition (NER) is the task of spotting mentions of named entities in text without linking them to canonical entities. NER involves classifying mentions into classes such as PERSON, ORGANIZATION, LOCATION and EVENT.

Named Entity Classification (NEC) is the problem of inferring semantic type labels for mentions of named entities in natural language text. Types can be coarse-grained such as PERSONS, LOCATIONS, ORGANIZATION or EVENTS, or fine-grained such as SCIENTISTS, EDUCATIONAL INSTITUTION or NATURAL CATASTROPHE.

Named Entity Disambiguation (NED) is the process of linking already annotated mentions of named entities to canonical entities registered in a Knowledge Base (KB). The problem is also known in the literature as entity linking.

Named Entity Recognition and Disambiguation (NERD) refers to the complete pipeline that does both NER, NEC and NED. It takes as input a natural language text without any mention annotations. As output, NERD systems yield both mention boundaries and links to canonical entities in a KB.

Entity Catalog is the part of the KB that contains a collection of uniquely identified canonical entities.

Name-Entity Dictionary is a many-to-many relation between surface forms and canonical entities. It contains potential names of entities, and hence it can be used to retrieve a list of candidate entities that a surface form can denote.

Entity Keyphrases is a set of characteristic phrases that describe a named entity.

Contextual Similarity is a measure for how similar an input document is to the description of an entity. It is estimated by comparing a statistical model for entity keyphrases against the input text by measures such as cosine similarity, KL divergence or weighted Jaccard distance.

Inverse Document Frequency (IDF) is a numerical score that reflects how specific or generic a word is. IDF value decreases proportionally to the number of documents that contain it. The more the documents that contain the word, the less characteristic it is.

Mutual Information (MI) between a keyword and an entity is a measure for how related or dependent they are, and how much knowing one of them increases the probability of predicting the other.

Entity-Entity Semantic Coherence is measure for semantic similarity or relatedness between named entities. For example, `David_Beckham` is more semantically coherent with `Manchester_United` club than with the `White_House`.

NERD Repository NERD requires an entity catalog, a name-entity dictionary, entity keyphrases and entity-entity coherence measures to solve the NERD problem. Therefore, NERD systems leverage data available in a KB to obtain these data components. However, preprocessing steps are often needed such as noise removal and statistics computation. The output of the preprocessing is stored in a NERD repository.

1.3. Challenges

Building a domain-specific NERD solution cannot be achieved by solely replacing the underlying KB. Similarly, supporting different languages does not merely entail translating all underlying text-oriented resources (e.g. entity name dictionaries). A truly versatile NERD system should be designed carefully in order to allow for replacing different NERD components. Below are the challenges that a customizable NERD architecture should address.

Multi-domain Text: Applying NED to general text such as news articles can be performed against general knowledge bases such as YAGO. However, domain-specific texts such as text about movies, music or books require specialized KBs such as IMDb, last.fm or LibraryThing, respectively. Furthermore, such text may span more than one domain and may also include entities that are captured only by general-purpose KBs. It is therefore mandatory to build an NED framework that is capable of disambiguating text against one or more knowledge bases collectively. The NERD architecture should address challenges that arise from combining different KBs that have potentially different schemas or different entity identification systems.

Limited KB Coverage: The union of KBs is a good first attempt to build an application-specific KB. But in realistic scenarios it is often required to combine different parts of different KBs. For example, consider the case of combining Wikipedia together with IMDb to build a NERD system geared for text about movies. IMDb has a very good coverage of actors and movies, but so does Wikipedia. However, since Wikipedia is not a perfect KB, it is better to ignore all movie entities in Wikipedia, and include only non-movie related entities. Therefore, a flexible architecture should allow for combining only parts of KBs together, ideally based on entity semantic types.

Combining Multiple KBs: Many prominent entities exist in multiple KBs. Therefore, combining different KBs introduces the risk of having redundant entities in the final entity repository. While it is not the focus of this work to detect equivalent entities across KBs, the architecture of the NERD framework should take this situation into consideration. Furthermore, the architecture should utilize the information about shared entities to build a consistent and comprehensive NERD system.

NERD Repository: State-of-the-art NED techniques require various data. This includes a mention-entity dictionary and a catalog of entity descriptions. Approaches that perform

a collective disambiguation of entities require an additional notion of entity-entity relatedness, which is usually computed using co-occurrence statistics. Those data components together form the entity repository, and are necessary for performing the NERD task. Wikipedia is a rich resource that classifies entities into categories, contains anchor links across articles, and provides dictionaries in the form of disambiguation pages and redirects. Together, these features facilitate the extraction of the data required for the NED task. Many other KBs are less expressive in that aspect, and many do not capture entity occurrences. A universal NERD architecture should be flexible enough to consume data from KBs and build dictionaries and contextual descriptions of entities. It should allow extracting keyphrases from free text, or utilize non-entity occurrence-based methods to estimate the relatedness between entities.

Type Hierarchies of KBs: Each KB organizes its entities under a type hierarchy. Unfortunately, there is no universal type hierarchy under which all entities across KBs are organized. NERD architectures should allow entities classified under different hierarchies to co-exist in the NERD schema. Furthermore, multiple type hierarchies should be supported during the NERD process.

Scarce Resources for non-English Languages: NERD requires comprehensive and precise resources that assist its machinery towards finding the correct mapping of mentions to entities. Examples for resources are name dictionaries of potential surface forms that can denote an entity. NED also requires characteristic description. Since such resources can be extracted from Wikipedia, they are relatively rich for the English language. For other languages, however, they are less comprehensive. Furthermore, for languages such as Arabic, for which Wikipedia has an order of magnitude smaller size than the English one, the available resource is far from complete and harms the quality of the NED process. It is mandatory to exploit existing English resources to enrich NERD methods for non-English languages.

Fine-grained Entity Type Classification: Knowledge bases organize entities under type taxonomies that are more fine-grained than the classical `PERSON`, `LOCATION`, `ORGANIZATION`, `EVENT`, `MISC` classification. Classifying entities to fine-grained taxonomies that distinguish between football players and basketball players (with both being athletes) is a challenging problem. For such sophisticated classification, machine-learning is a promising option.

1.4. Contributions

The contributions of this dissertation are focused on the problem of building a universally applicable and customizable NERD framework that supports processing text from different domains and languages. This work addresses that problem by introducing the U-AIDA architecture. U-AIDA is named after AIDA[60], a state-of-the-art NERD system. "U" serves to denote a double meaning: it signals the fact that this is a **universal** architecture suitable for different domains and languages, and "U" is also interpreted as **your** AIDA, indicating that you are able to customize it to build your own NERD system that fits your application scenario.

The contributions can be summarized as follows:

U-AIDA: A Universal Architecture for Building NERD Solutions. We developed U-AIDA, a universal and flexible NERD architecture that supports plugging in different KBs and components. U-AIDA supports customization to fit different application scenarios and handle text covering various domains.

Domain-Specific NERD Solution. We utilized U-AIDA to build a domain-specific NERD solution for disambiguating documents from the German National Library (DNB). Our solution combines the YAGO KB together with the KB developed by DNB. In addition, we exploited U-AIDA to build a custom NERD solution geared towards social streams. Using the system developed for social streams, we participated in the #Microposts2014 Challenge [61].

NERD for Languages with Poor Entity-Resources. We designed and implemented a complete pipeline for building a NERD solution for languages other than English, incorporating resource enrichment techniques. The pipeline has been tested for three languages (Spanish, Italian and Arabic) and can be applied to other languages. The system has been implemented within the U-AIDA framework. A first version without data enrichment techniques focused on Arabic and has been published in the ANLP 2014 EMNLP Workshop [62]. The complete pipeline has been published in the ESAIR 2015 CIKM Workshop [20].

HYENA System for NEC. We developed a machine learning based approach to classify mentions of named entities in natural language text under a fine-grained type hierarchy of 505 semantic classes. This work has been published in the COLING 2012 Conference [58] and demonstrated in the ACL 2013 Conference [59].

1.5. Organization

The rest of the thesis is organized as follows. Chapter 2 gives the general background about the problem of NERD and state-of-the-art NERD systems. Chapter 3 explains the customizable U-AIDA architecture for NERD systems. In Chapter 4, we discuss the problem of building a domain specific NED solution within the proposed framework. Chapter 5 discusses the challenges of applying NED for languages with scarce resources such as Arabic, and our proposal to automatically enrich such resources. A machine-learning approach for entity type classification is presented in Chapter 6. We finally make concluding remarks in Chapter 7.

2. Background and Related Work

2.1. Knowledge Bases

The general definition of a *knowledge base* (KB) is a set of facts and rules stored in a machine readable format. KBs enable computers to solve problems that require background information. For example, to build a medical system for automatic diagnosis, a KB about different diseases and their symptoms is required. Natural language question-answering systems depend on a KB of general knowledge to retrieve the answers of different questions.

Wikipedia is the most prominent general-purpose knowledge resource available publicly. It contains information about Named Entities such as `PERSONS`, `LOCATIONS`, `ORGANIZATIONS` and `EVENTS`. In addition, there are articles about general concepts such as the definition of a "Programming Language", or the theory behind how airplanes work.

Most of the data in Wikipedia is in the form of natural language text. In addition, it has structured data such as infoboxes, the category system, and the graph representing the links in Wikipedia.

Many knowledge bases, such as YAGO [25], are derived from Wikipedia. Such KBs are more machine-friendly in the sense that they store information in a formal representation composed of semantic relations between entities. For example,

```
<Albert_Einstein> wasBornIn <Ulm>
```

is a fact between a `PERSON` and a `CITY` using the semantic relation `wasBornIn`. In addition, many KBs organize entities under a hierarchy of semantic types, and store this information in the form of facts as well:

```
<Albert_Einstein> isA <Physicist>
```

KBs are abundantly used in online services. Google has a Knowledge Graph that helps the company to improve the quality of its services such as search and recommendations.

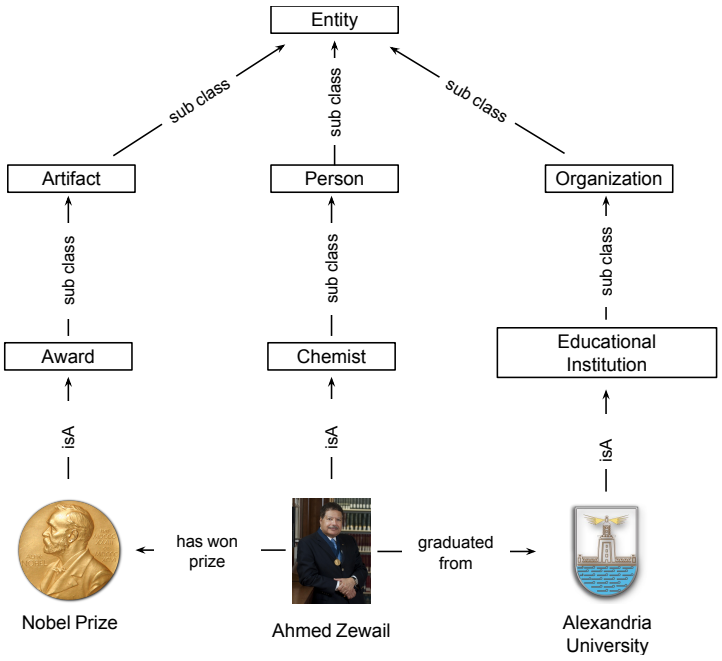

Figure 2.1.: Example from YAGO Knowledge Base

Similarly, Apple and Microsoft leverage unpublished KBs to enable their voice assistance services.

In addition to such general-purpose KBs, many specialized ones exist. For example, IMDb contains a comprehensive KB about the movie business. It contains entities about different movies, actors, directors, etc. Similarly, last.fm has a large KB about songs and artists. LibraryThing uses a KB that is built especially for books. The German National Library offers the GND[1]: its own KB with a huge collection of works, authors and publishing organizations for the German language.

While all these KBs share the property of storing data in the form of facts, they vary significantly in the semantics that they capture. For example, while IMDb has semantic relations such as `directedBy` and `producedIn`, LibraryThing uses relations such as `hasAuthor` and `hasCitation`.

Every KB comes with a schema that defines the set of relations between entities, as well as the type hierarchy under which entities are classified. Figure 2.1 depicts a snapshot of the YAGO KB.

[1] `http://www.dnb.de/gnd`

2.2. Named-Entity Recognition and Disambiguation

Mentions of named entities such as people, places, organizations, etc. appear frequently in sources such as web pages, news articles, and other Internet content. Most of the names have more than one potential meaning, and are therefore ambiguous. Consider this example:

> Paris is a nice city

The mention "Paris" might refer to the French capital, or a small city in the US. Nevertheless, with so little context, it mostly refers to the French capital, since it is way more prominent. This is an easy example. Consider the following example:

> A new book about Paris has been published

The mention "Paris" here does not necessarily denote a city; it may refer to the famous character in the Greek mythology. Hence the problem is more complex and requires careful consideration for other data ingredients like the input context.

Named Entity Recognition and Disambiguation (NERD) is the problem of spotting mentions of named entities in an input text and mapping them onto canonical entities registered in a Knowledge Base. Consider a more complex example:

> Mike and his colleagues Rowe and Wong were the architects of the relational system Ingres developed at Berkeley

Going for the most prominent entity will wrongly map the mention "Berkeley" to the American city instead of the university. Similarly, we should be able to tell that "Mike" refers to the computer scientist Michael Stonebraker and not to the tennis player Mike Bryan or the pop singer Michael Jackson, and that "Ingres" is a database system rather than the french painter Jean-Auguste-Dominique Ingres.

The key here is to consider the *context* of the mention to be mapped, and compare it - by some *similarity measure* - to contextual information about the potential target entities. Given our example sentence, clues like "architects", and "developed" should be

considered when building the mention context, and guide the system to pick entities that are computer related because such entities will potentially have similar *keywords* in their description in contrast to other candidates. In other words, we can build a sequence-of-words model for the input text, and another model for each candidate entity, built on top of the characteristic words in their Wikipedia articles, for example. By comparing both models (using measures such as cosine similarity, weighted Jaccard distance or KL divergence), we can pick the candidate entity with the highest similarity score to be the correct entity. Other approaches such as training a multi-way classifier from labeled training data, or exploiting part-of-speech tags or dependency-parsing paths can also be used to address the Named Entity Disambiguation (NED) problem.

The previously discussed approaches are local in the sense that mentions are disambiguated one at a time. Local methods might work well for sufficiently long and relatively clean input text, such as predicting the link target of a Wikipedia anchor text [39]. However, for less clean text such as arbitrary Web pages, or shorter text such as microblogs, relying only on contextual similarity will not yield high disambiguation quality.

One last key ingredient to further improve the disambiguation quality is to incorporate a joint disambiguation model by *collectively* considering multiple mentions in the input. The model should consider the *semantic relatedness* among entities. Considering our example, "Mike", "Rowe", "Wong" and "Ingres" are mentions that should be collectively disambiguated together with "Berkeley". The underlying assumption here is that the text is thematically homogeneous in the sense that all entities mentioned in the text are semantically related or *coherent*.

In summary, there are three major approaches to perform the NED task.

- **Prominence-based** approaches that ignore the surrounding context of the mention, and assume that the most prominent candidate is the correct solution.
- **Local methods** that pick the entity with the characteristic description that is most similar to the input context of the mention.
- **Collective methods** that assume the input text is semantically coherent and hence take into account the semantic relatedness between entities.

The rest of this chapter discusses prior work on NED and NEC. Finally, we briefly explain AIDA[26], a state-of-the-art system for NERD, that incorporates all of the three ingredients to improve robustness.

2.3. NED: State-of-the-art

Several NED systems have been developed for the English language such as DBpedia Spotlight [38], Tagme2 [17], AIDA [26, 60], and Babelfy [40]. Only few of those systems are capable of processing input in other languages. Furthermore, up to our knowledge, only Babelfy has support for Arabic NED.

Babelfy is a multilingual system that combines both the Word Sense Disambiguation (WSD) and the NED tasks. They use BabelNet[43] as their underlying KB, and leverage machine translation to translate only Wikipedia concepts into several other languages. However, they excluded named entities from translation.

McNamee *et al.* [37] developed a cross-language entity linking approach. They used the English Wikipedia entities, extracted under TAC KBP [57], as their reference knowledge base. Their proposed solution was to transform the problem to a monolingual English problem. Therefore, they translated the input to English before applying NED. They also developed a persons-only cross-language ground truth to evaluate their approach, exploiting parallel corpora and crowd-sourcing [36]. However, applying NED for other languages cannot be performed by simply translating the problem to the English language. One issue is the errors due to the automatic machine translation. Another drawback is the entity repository. Text of language L mostly refers to entities that are specific to the region where this language is spoken. Therefore, it requires different entity set other than used when applying NED on English text.

Many research efforts have been conducted to enhance the quality of Named Entity (NE) translation. Huang *et al.* [28] introduced the usage of phonetic and semantic similarity in order to improve NEs translation. Azab *et al.* [4] developed a classification technique to decide whether to translate or transliterate named-entities when translating full text from English to Arabic. They used combination of the token-based, semantic and contextual features in the classification model. Also, Lee *et al.* [31] proposed including the part-of-speech tagging information in the translation process to enhance the translation of person names in text.

Furthermore, most of the existing systems disambiguate to Wikipedia or a KB built from it. AGDISTIS [54, 53] is a KB-agnostic NED framework, however, it has been tested against two general purpose KBs, YAGO [25] and DBpedia [3], and only one KB at a time.

2.4. Background and Prior Work on Named Entity Classification (NEC)

There is little prior work on the task of classifying named entities, given in the form of (still ambiguous) noun phrases, onto fine-grained lexical types. The following methods are also considered in our experiments as state-of-the-art baselines.

[19] has been the first work to address type granularities that are finer than the handful of tags used in classical NER work (person, organization, location, date, money, other – see, e.g., [55, 2, 10, 18]). It considered 8 sub-classes of the `Person` class, and developed a decision-tree classifier based on the following features: unigrams and bigrams in the mention and surrounding text; topic signatures derived from the general words occurring in mention contexts for each class and weighted by class-discriminative scores; and an expanded variant of the latter using words from WordNet synonyms and hypernyms, again with clever weighting.

[13] considered 141 subtypes of the WordNet class `Person`, and developed a maximum entropy classifier using word-level features from the mention contexts. In addition to the words themselves, these include POS tags, capitalization, word lengths, special tokens like digits, and Lesk-style expansion with WordNet glosses [32]. Their experimental results are flagged as non-reproducible in the ACL Anthology.

[52] considered a two-level type hierarchy consisting of 29 top-level classes and a total of 92 sub-classes. These include many non-entity types such as date, time, percent, money, quantity, ordinal, cardinal, etc. The method uses a rich set of features: separating context words from context verbs, prefixes and suffixes of compound noun phrases, tags given by the Stanford NER tagger, POS tags, capitalization, presence of words in eight different gazetteers for coarse-grained classes (person names, location names, etc.), and WordNet senses of noun-phrase head words in mention contexts. The latter required manual sense-tagging (evaluation is on an already sense-tagged corpus); however, the experiments in [52] showed that this kind of feature did not contribute well to accurate results. The feature representation of test instances was fed into a two-level hierarchical classifier – with the limitation that the classifier was designed to assign only one type label to each instance (for each of the two levels). The method employed a simple form of collective inference by enforcing that multiple identical mentions in the same input text are assigned the same type label: the one with the highest joint evidence.

[21] proposed an SVD-based latent topic model with a semantic kernel that captures word proximities. The method was applied to a set of 21 different types; each mention is assigned to exactly one type.

The very recent work of [34] considered a two-level taxonomy with 112 tags taken from

the Freebase knowledge base, forming a two-level hierarchy with top-level topics and 112 types (with entity instances). [34] trained a CRF for the joint task of recognizing entity mentions and inferring type tags. The feature set included the ones used in earlier work (see above) plus patterns from ReVerb [14]. This is the only prior work that could assign multiple type labels to the same mention. In the reported experiments, 562 entity mentions from 434 sentences were given a total of 771 tags – a modest amount of multi-label tagging.

2.5. The AIDA System

2.5.1. AIDA in a Nutshell

Solving the NERD problem is crucial to enable harvesting knowledge from large data and text collections [56]. Therefore, many approaches have been introduced in the literature that address this problem. AIDA is a NERD system that leverages knowledge bases (such as YAGO) as a repository of entities classified under a semantic type hierarchy. In addition, AIDA exploits the semantic relations among entities to compute similarity and coherence measures that are integrated into a collective graph-based disambiguation approach. AIDA uses the English Stanford Named Entity Recognizer [18] to annotate mentions of named entities in the input text.

AIDA casts the disambiguation problem into a graph. The graph contains two types of nodes: mention nodes and entity nodes, and two types of edges, edges between a mention and a candidate entity (*m-e edges*), and edges between entities (*e-e edges*). All edges in the graph are weighted where *m-e* edge weights capture the contextual similarity between the input text and the candidate entity, and *e-e* edge weights capture the semantic relatedness or the coherence between entities. The goal is to identify a dense sub-graph that contains *exactly one* candidate entity connected to *every mention*, yielding the most likely problem solution.

Figure 2.2 illustrates an example mention-entity graph for an input text with highlighted mentions (left) and candidate entities (middle) based on a knowledge base (right). The thickness of edges between entities depicts different edge weights.

2.5.2. Data and Measures

Entity Candidates: In order to build the list of potential candidates denoted by a mention, AIDA uses the YAGO knowledge base to retrieve the list of potential entities denoted by a name. Entity names in YAGO are available via the `means` relation which in turn is extracted from Wikipedia disambiguation pages, redirects and anchor texts.

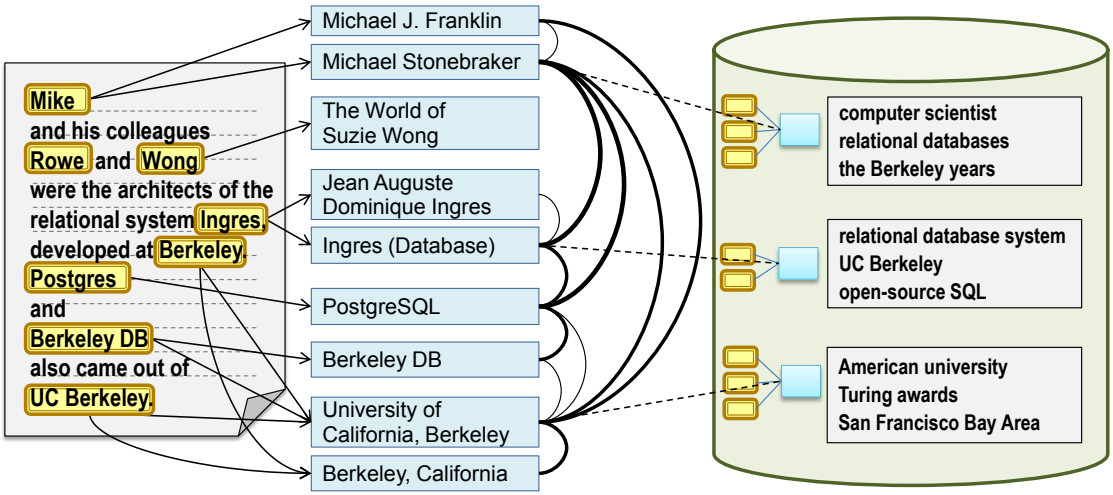

Figure 2.2.: Example for AIDA Mention-Entity Graph

Entity Prominence: AIDA estimates entity prominence using a probabilistic prior for mapping a name to an entity. AIDA computes this probability via Wikipedia-based frequencies of names in link anchor texts referring to specific entities. The output for this is an estimate for the most natural addressed entity for a specific mention. For example, "Berkeley" refers to `Berkeley` (the city) in 62.6% of all occurrences and in 21.4% to `University of California, Berkeley`.

Context Similarity: AIDA builds a sequence-of-words representation of the input text that is considered as the mention's context. On the entity side of the mapping, each entity is associated with a set of characteristic phrases or salient words. Entity keyphrases are harvested from anchor texts in Wikipedia articles. In addition, Wikipedia category names as well as titles of pages with incoming links are added to the keyphrase set of an entity.

Tokens of a keyphrase are refereed to as keywords. Keyphrase and keywords are assigned *global specificity weights* based on their global prominence in Wikipedia. Global specific scores of keywords are estimated using *Inverse Document Frequency* (IDF) scores where each entity is considered a document, and a keyword belongs to an entity if it exists in its set of keyphrases. More specifically, it is defined as:

$$IDF(w) = \frac{|\{e : w \in KP(e)\}|}{N}$$

where $KP(e)$ is the set of keyphrases of an entity e, and N denotes the total number of entities. Global specificity score of a keyphase is defined to be the summation of scores of its keywords.

In addition, keywords are assigned *per-entity specificity weights* based on the *Mutual Information* (MI) between an entity and a keyword. Mutual Information between an entity e and a keyword w is defined as:

$$MI(e;w) = \sum_{y \in \{W, \bar{W}\}} \sum_{x \in \{E, \bar{E}\}} p(x,y) \log \left(\frac{p(x,y)}{p(x)\, p(y)} \right)$$

where W (or $\bar{W}$) denotes the event that a keyword w occurs (or does not occur) in a document. Similarly, E (or $\bar{E}$) denotes that an entity e occurs (or does not occur) in a document. A document is defined as a Wikipedia page of an entity. A keyword w occurs in an entity document if it matches any of the tokens of the entity keyphrases. An entity e occurs in an entity document if there is an outgoing link from that entity Wikipedia page to e. The co-occurrence probability between an entity e and a keyword w is calculated as follows:

$$p(e \wedge w) = \frac{|\{e' \in \{e \cup IN_e\} : w \in KP(e')\}|}{N}$$

where $KP(e)$ is the set of keyphrases of an entity e. This reflects how frequently w appears in the keyphrase set of e or any of the keyphrase sets of an entity linking to e, $IN(e)$, with N denoting the total number of entities. The joint probabilities for the cases $p(e, \bar{w})$, $p(\bar{e}, w)$, $p(\bar{e}, \bar{w})$ are calculated accordingly.

MI scores are computed between entities and keywords instead of keyphrases. The reason is that AIDA uses a weighted partial matching model between the input text and the entity context to compute the contextual similarity between a mention and one of its candidate entities [26].

Coherence among Entities: AIDA estimates the semantic relatedness between entities using the Wikipeida link structure. The more frequent two entities co-occur in Wikipedia, the higher their semantic relatedness score should be. Therefore, AIDA estimates the entity-entity coherence using the inlink overlap by the approach refined by Milne and Witten [39] that takes into account the total number N of entities as follows:

$$mw\_coh(e_1, e_2) =$$
$$1 - \frac{\log\left(\max(|IN_{e_1}|, |IN_{e_2}|)\right) - \log(|IN_{e_1} \cap IN_{e_2}|)}{\log(N) - \log\left(\min(|IN_{e_1}|, |IN_{e_2}|)\right)}$$

where IN_e is the set of entities linking to an entity e.

Overall Objective Function: AIDA combines the prominence score, contextual similarity and semantic coherence to build a combined objective function as follows: For each mention m_i, $i = 1..k$, select entity candidates e_{j_i}, one per mention, such that

$$\alpha \cdot \sum_{i=1..k} \text{prior}(m_i, e_{j_i}) +$$
$$\beta \cdot \sum_{i=1..k} \text{sim}(\text{cxt}(m_i), \text{cxt}(e_{j_i})) +$$
$$\gamma \cdot \text{coh}(e_{j_1} \in \text{cnd}(m_1) \ldots e_{j_k} \in \text{cnd}(m_k)) = \max!$$

where $\alpha + \beta + \gamma = 1$, $\text{cnd}(m_i)$ is the set of possible meanings of m_i, $\text{cxt}(.)$ denotes the context of mentions and entities, respectively, and $\text{coh}(.)$ is the coherence function for a set of entities.

2.5.3. Model and Algorithm

AIDA casts the NED task to a graph problem by constructing a weighted undirected graph with mentions and candidate entities as nodes. Edges between mentions and entities are weighted with a linear combination of prominence score and similarity measure. Edges between entities are weighted based on the Wikipedia-link overlap.

The objective of AIDA is to compute a dense sub-graph that contains all mention nodes and exactly one mention-entity edge per mention. In order to achieve high accuracy for the long tail of less prominent entities by capturing the weak links in the overall graph, it defines the density of the sub-graph to be the minimum weighted degree among its nodes, where the weighted degree of a node is the total weight of its incident edges. The final objective of the algorithm is to find the subgraph with the highest minimum weighted degree satisfying the one-entity-per-mention constraint. AIDA adapts an approximation algorithm to solve the dense subgraph problem.

For mentions that denote entities that do not exist in the underlying KB, AIDA adds a per-mention placeholder Out-of-KB entity node in the graph. The keyphrase model for such nodes is extracted from the web [23]. In addition, AIDA applies a keyphrase-based entity-entity relatedness measure [24].

3. U-AIDA Architecture of a Customizable NERD Framework

3.1. Overview

Building a Named Entity Recognition and Disambiguation (NERD) system that is capable of processing inputs from different text genres and languages requires a flexible architecture that supports adding and removing building blocks easily. For example, processing a piece of text from a special domain cannot be performed by solely replacing the knowledge base. Instead, it entails many other design changes.

Different NER techniques are suitable for different genres of text. For example, a Named Entity Recognition (NER) system developed and optimized for recognizing mentions in news articles will perform poorly on user-generated content in social media such as Twitter.

Furthermore, given a specific domain, different application scenarios entail different processing approaches. For example, a NERD system that is geared for high recall will apply NERD techniques differently from those optimized for precision. A truly flexible architecture should allow such on-the-fly configurations.

In addition, different kinds of input text require different input representation. For example, disambiguating a 140-character tweet is handled differently from a 100-page book. Each requires optimized representations and techniques to obtain the best trade off between quality and performance.

The rest of this chapter is dedicated to explaining how AIDA is architected such that it supports processing documents from different domains and languages and of different styles. The whole NERD pipeline is revisited and each stage is discussed in details.

3.2. Named Entity Recognition

This section addresses the following research issues:

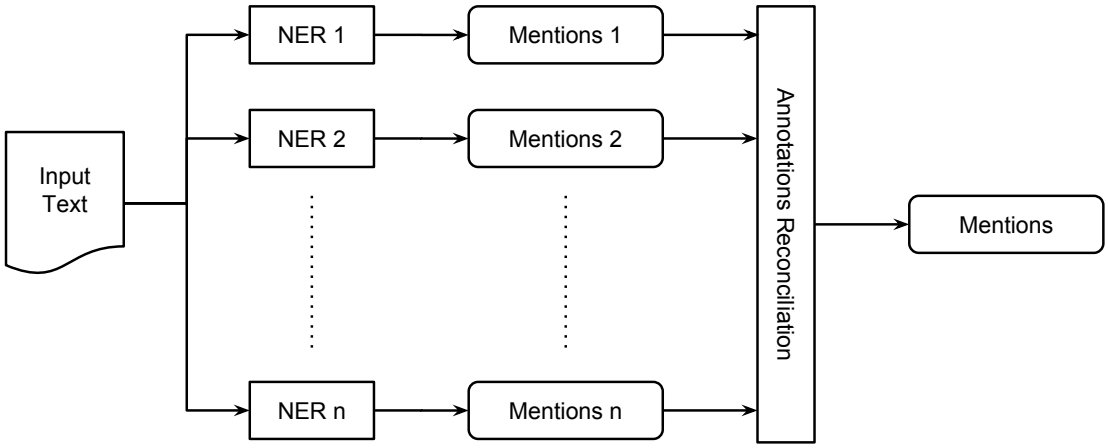

Figure 3.1.: The Architecture of the U-AIDA NER Component

- Mention recognition is language-specific.
- Different text genres require different NER techniques.
- For high recall, we need to invoke multiple NER methods and combine their outputs.

NER is Language-Specific. The very first stage in a NERD system is to spot the mentions of named entities in the input text. NER is a language-specific component. State-of-the-art NER systems developed for the English languages utilize many linguistic features such as capitalization. For languages such as Arabic (which lacks any notion of capitalization) or German (where capitalization denotes another interpretation), different NER techniques should be applied. In other words, an NER system developed and optimized for the English language is not directly applicable to Arabic or German for example. Furthermore, many special domains (such as the medical domain) needs custom-made NER systems that incorporate comprehensive dictionaries of named entities. U-AIDA addresses this by isolating the NER component from the rest of pipeline. Accordingly, for different applications scenarios, different NER systems are integrated in the NERD solution.

NER for Different Text Genres. In addition to language and domain, text style influences the NER task. NER systems geared for polished texts such as news articles will perform poorly on less formal texts such as blog posts, in which for instance grammatical mistake are more likely to occur. Dealing with different text styles requires non-standard NER techniques.

For example, microposts from social streams include extra clues for recognizing mentions of named entities. Considering Twitter as an example, tweets use the character '@' to refer to other Twitter users. While such users may or may not exist in our entity repository, the '@' character is a strong clue that the following is a named entity (e.g. a person or organization). Similarly, the '#' characters denotes a *hashtag* which mostly refer to a specific event or a person. Hence, an NER system developed for Twitter or a similar social stream should handle such characters differently.

Given the flexibility of the U-AIDA architecture, it is possible to plug in a custom NER technique without affecting the rest of the pipeline. Our custom-made NER component developed for Twitter is discussed in Section 4.2.

U-AIDA provides a suite of different NER techniques suitable for most of the standard use cases including Stanford NER [18]. Depending on the application scenario, U-AIDA can be configured to invoke one of provided NER techniques, or a custom one can be plugged into the NERD pipeline.

Multiple NER Systems are Required Together. There exist multiple NER techniques even for the same text language and genre. For example, Stanford NER tagger [18], Apache OpenNLP[1], Illinois NETager [44] and GATE [11, 10] are all capable of recognizing mentions of named entities in English text. However, they do not necessarily output the same set of annotations. When different taggers are run on the same input text, their output annotations can be combined together to produce annotations with higher precision or higher recall.

As depicted in Figure 3.1, U-AIDA takes as an input a configurable set of NER taggers and invokes all of them on the input text. By default, U-AIDA is geared for high recall when compiling the final set of mention annotations by taking the union of all annotations produced by different NER taggers. U-AIDA reconciles overlapping annotations by merging them into the longest possible mention. For example, given this sentence:

> Chancellor Angela Merkel and President Barack Obama are closely monitoring the Greek debt crisis

[1] `http://opennlp.apache.org/`

if one tagger produces the following mentions "Angela Merkel", "Barack Obama" and "Greek debt crisis", and another annotates "Chancellor Angela Merkel", "President Barack Obama" and "Greek", the final set of mentions will be reconciled as "Chancellor Angela Merkel", "President Barack Obama" and "Greek debt crisis".

3.3. Input Text Representation

This section addresses the following research issues:

- For some input genres, mentions can be represented in forms other than what explicitly appears in the input text.
- For some application scenarios, input context should be expanded using background knowledge.

The standard input to a NERD system is a natural language text represented as a sequence of tokens. The input text is fed into the NER stage to spot the mentions of named entities that are provided to other stages in the NERD pipeline. Depending on the nature and language of the input text, a preprocessor is invoked to do proper text normalization and segmentation. In Arabic for example, it is crucial to remove the "Kashida" elongation character used to stretch the text to align both the left and right ends of the line (in contrast to white-space being the elongation character in English). In addition, languages that incorporate a nature of clitics[2] should be segmented carefully. For example, proclitics (such as "je **t**'aime" in French) or enclitics (such as "Olen kiinnostunut kvanttifysiika**sta**" in Finnish) should be properly segmented from their host words. Arabic is also very rich language in terms of clitics.

While such representation of the input as a sequence of tokens and a set of mentions is suitable for the general case, it requires extensions for more specialized cases. Our architecture supports extending the representation of both the input text as well as the set of mentions as explained below.

Mentions. A mention is a span in the input text that contains one or more tokens that together denote a named entity. For example, the following are all mentions that (may) refer to the same entity `Barack_Obama`: "Obama", "Barack Obama", "Mr. Obama",

[2] `https://en.wikipedia.org/wiki/Clitic`

"President Obama", "Barack Huessin Obama". In each case the mention string is used to query the entity names dictionary to build the list of candidate entities. But such an approach is not suitable for all application scenarios.

Consider Twitter for example. As explained earlier, *hashtags* and Twitter *user mentions* are good candidates for named entities. Consider the following tweet:

> U.S. Team and @SeattleReignFC midfielder @mPinoe throws out ceremonial pitch at #Mariners game. #WorldCup2015

Applying the standard mention representation would result in the following mention "#WorldCup2015". This representation contains an extra '#' character which does not exist in the name entity dictionary. Furthermore, another form can be automatically extracted by removing the '#' character and splitting it based on capitalization. This will suggest "World Cup 2015" as a potential mention which is more suitable when querying the name entity dictionary. However, some names should not be split such as the English lexical database "WordNet" and the music album "The ConstruKction of Light"[3]. Therefore, before actually solving the disambiguation problem, it is not possible to decide whether the name should be split or not. Hence, at this stage, both forms (split and non-split) are potential mentions.

U-AIDA addresses this problem by associating a set of potential mention representations with each explicit mention. All implicit mentions are used when querying the dictionary to build the list of the candidate entities. More details are discussed in the Twitter use case in Section 4.2.

Input Context. The context of a mention is in general the sequence of its surrounding tokens. It is crucial to get the context right because this is the key for all similarity-based disambiguation methods. Considering all surrounding tokens as the mentions context is suitable for the general case such as news article. But building a flexible NERD framework that is capable of processing text from different sources requires going beyond the surrounding tokens. Namely, the framework should support expanding the mention context beyond the input text, and such context extension should be dealt with in the same way as the original input context apart from the fact that it does not go through the NER stage.

[3] `https://en.wikipedia.org/wiki/The_ConstruKction_of_Light`

To justify the usefulness of this form of context expansion, let us consider two examples. The first is Twitter, where it is very challenging to correctly disambiguate mentions of named entities given the 140-character long input context alone. Many approaches in the literature such as [29] incorporate context extensions by clustering tweets, resolving URLs in the tweet, and leveraging hashtag definition gazetteers. Another example, is to disambiguate a whole book. It is infeasible to consider the whole book as the input context, mainly for performance reasons. However, the introduction section or the first paragraph of every chapter might be useful when disambiguating other passages in the book. It can be useful to extend the context of a chapter with the first paragraph of other chapters, for example.

To this end, the architecture should support extending the input context by other text passages. Such an extended context is provided for other stages in the pipeline to enhance similarity-based measures. It is up to the specific application to decide on the context extension technique it adopts.

3.4. Knowledge Base

The Knowledge base is one of the most important and influential components in any NERD system. It contains the entity catalog and hence determines the set of entities a NERD framework can assign mentions to. In addition, it provides most of the data required for performing the NERD. A knowledge base contains the name-entity dictionary, entity-characteristic keyphrases, and potentially other data required to compute prominence scores for entities and specificity scores for keywords. Furthermore, the KB has the statistics to estimate the semantic relatedness between entities.

Replacing the knowledge base does not merely imply plugging in another set of facts into the NERD framework where relations have different names. A flexible system should consider all potential variations in providing each of the required data ingredients and provide the downstream applications the support for turning different data pieces ON or OFF during the disambiguation process. The rest of this section discusses in detail each of the data components and what design decisions we have taken in order to support as much customizability as possible.

3.4.1. Entity Repository

This section addresses the following research issues:

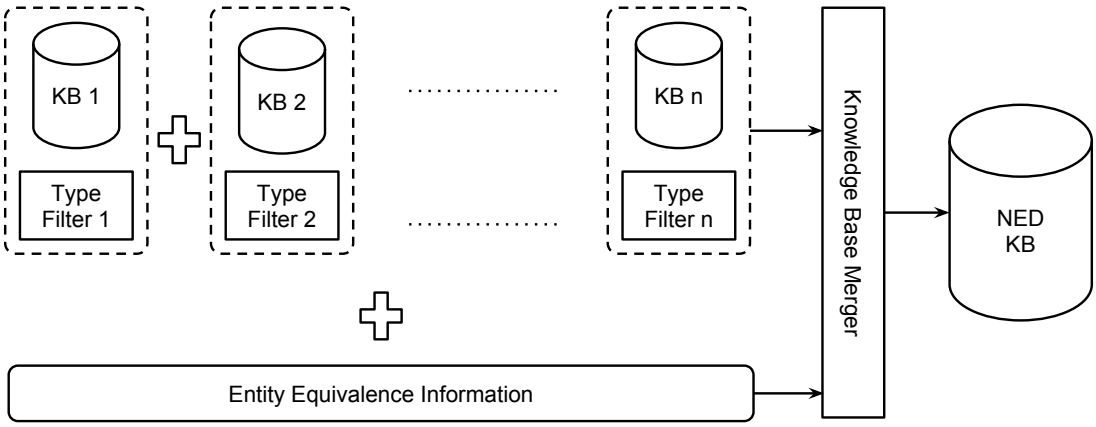

Figure 3.2.: U-AIDA Approach for Combining Knowledge Bases

- It may be necessary to disambiguate against multiple knowledge bases.
- Knowledge bases share entities.
- Entities in different knowledge bases are organized under different type. hierarchies.

NED against Multiple Knowledge Bases. The general case for building a NERD system assumes a comprehensive underlying knowledge base where all entities exist. Such an approach places most of the critical work on the knowledge base building stage. For example, to disambiguate against entities from Wikipedia and IMDb, it is first necessary to build one large entity repository that contains the union of entities existing in each resource. Combining entity catalogs cannot be done by simple aggregation. Many entities exist in more than one KB. Therefore, simple aggregation will produce entity repository that contains many redundant entities.

The U-AIDA architecture takes care of that by merging all equivalent entities into one canonical entity. U-AIDA assumes that entity equivalence information is already provided as an input, but it leverages such information to avoid any entity redundancy.

Different KBs exhibit different Type Hierarchies. Most of the knowledge bases organize entities under a semantic type hierarchy. Each knowledge base defines its own type

hierarchy, which is not necessarily compatible with other type hierarchies defined by other knowledge bases.

The U-AIDA architecture allows entities originated from different knowledge bases to be classified under different type hierarchies. U-AIDA does not make any assumptions about the input knowledge bases. U-AIDA achieves that by fully identifying semantic types taking into account the source KB. Considering IMDb and Wikipedia example, both knowledge resources contain a semantic type for movies. U-AIDA distinguishes between both types by their source. Hence, a movie entity originating from Wikipedia will be assigned the type "`WIKIPEDIA:MOVIE`" and a movie from IMDb will be tagged with "`IMDB:MOVIE`". This provides downstream applications with complete control over the entity repository and eliminates any side effects of combining incompatible type hierarchies.

3.4.2. Entity Global Prominence

This section addresses the following research issues:

- For different knowledge bases, there exist different approaches to estimate entity prominence.
- For good quality, different entity prominence scores should be combined to produce one prominence score.

One of the simplest yet most influential clues for NERD is entity prominence. "Barack" almost always denotes the US president, and "Paris" denotes the French capital. Entity prominence can be estimated using many different ways. One is to consider Wikipedia page lengths, page update frequencies, the numbers of incoming or outgoing links, or the existence in other Wikipedia languages. For applications that process up-to-date information such as tweets, it has been shown that time-varying prominence estimation delivers the best quality [29]. Such time-varying prominence can be estimated from Wikipedia page view counts, for example.

It is rarely the case that one prominence measure is universally appropriate. U-AIDA takes as an input a suite of entity prominence measures. The final prominence score is computed via a configurable linear combination of all available prominence scores using a weight function that can be determined at run-time.

3.4.3. Name-Entity Dictionary

This section addresses the following research issues:

- Entity names are extracted from different sources of different qualities.
- Type-based candidate filtering is crucial.

The name-entity dictionary is key to building the entity candidate list for applying NERD to an input document. A noisy dictionary will add many wrong candidates that will make the NERD problem unnecessarily harder. On the other hand, an incomplete dictionary will miss potentially correct candidates.

U-AIDA has a carefully designed component for querying the dictionary. In order to build the list of candidates entities for a mention, U-AIDA requires three pieces of data:

1. **A Set of Surface Forms**: set of potential mention presentation including the original mention string.
2. **Types White List**: list of potential semantic types of a mention. Only entities belonging to one of those types are retrieved.
3. **Sources White List**: list of name sources. Only entries that are extracted from one of those sources are retired.

The retrieved **name-entity** entries must satisfy the following three criterion: the name exists in the set of *surface forms*, the entity is typed with one of the semantic types in the *types white list*, and the entry has been extracted using one of the extraction sources in the *sources while list*.

Each of the input data plays an important role in building a universal candidate entity retrieval component. As explained in Section 3.3, U-AIDA associates each mention with a set of potential representations that are compiled in the set of **surface forms**. In addition, U-AIDA allows per-mention **type white list**. For example, for a mention that is known to be a person (using techniques such as HYENA introduced in Chapter 6) it makes sense to exclude non-person entities from the candidate list. For example, for a mention "Gate" that has been tagged as a person by some entity type classification system, we should exclude the entity `GATES_FOUNDATION` from the entity candidate list.

Furthermore, typically, entries in the dictionary originate from multiple sources such as Wikipedia redirects or disambiguation pages. Different dictionary entry sources exhibit

different levels of quality. For example, Wikipedia page titles are on average of better quality than anchor texts. Based on the application scenario, only subset of those sources are considered using the **sources white list**.

3.4.4. Entity-Characteristic Keyphrases

This section addresses the following research issues:

- According to the underlying knowledge base, different techniques are incorporated to extract keyphrases.
- Keyphrases are extracted from various sources of different qualities.
- According to the application scenario, sources should be penalized or totally discarded.

Once a list of candidate entities is extracted, the next step is to find the most similar entity to the input context. This is performed by building a textual representation of each of the potential target entities, and comparing it to the input context. Depending on the underlying knowledge base, different data ingredients are included in the set of entity characteristic phrases. For example, for a well-structured and organized knowledge resource such as Wikipedia, category names as well as anchor texts of an entity are potential keyphrases that contribute to the textual representation of an entity. For knowledge bases such as Freebase, semantic relations such as "`graduated_from`", "`capital_of`" and "`acted_in`" are potential relations for building entities context. In addition, keyphrases can also be extracted from natural language text such as the first paragraph in Wikipedia pages.

Keyphrases are extracted differently from various knowledge bases. U-AIDA gives downstream applications full control on deciding what pieces of data are suitable to be included in the keyphrases set. Therefore, the U-AIDA architecture does not impose any hard assumptions on keyphrases extractions. It adopts a generic definition of entity keyphrases to be a list of phrases. Downstream applications are required to provide keyphrases extractors that output per-entity list of phrases.

In principle, any keyphrase extractor can be plugged into the pipeline. U-AIDA is equipped with extractors capable of processing Wikipedia and extract structured information from YAGO. In addition, U-AIDA contains an extractor that goes beyond

structured data and extracts keyphrases from natural language text. It applies a pattern-based extraction on POS tags to extract all noun phrases. It is easy to plug in further types of extractors.

Keyphrases - as well as keywords - are assigned *Inverse Document Frequency* (IDF) and *Mutual Information* (MI) scores (or weights) that reflect their specificity and correlation with different entities, respectively. U-AIDA uses the same score computation techniques as the original AIDA system that are detailed in Section 2.5.

Computing MI scores requires co-occurrence statistics between entities and keywords. To this end, we need a corpus with document level entity annotations. Wikipedia articles, for example, are annotated with hyperlinks to other entities. Hence, it is possible to compute how frequently an entity and a keyword co-occur in the same article. If the underlying knowledge base does not provide any notion of entity occurrence in a corpus, U-AIDA computes only global IDF score for the different keywords. During the disambiguation process, U-AIDA exploits whatever scores it could compute to deliver the best possible disambiguation quality.

Different keyphrase sources are of different qualities. In general, keyphrases are extracted from multiple sources. Consider Wikipedia for example: keyphrase are extracted from anchor texts, citation titles, Wikipedia category names, and titles of pages with incoming links. Usually, the first two are of lower quality than the rest, and hence some could be ignored depending on the precision-recall trade-off requirements. To provide the maximum flexibility to downstream applications, U-AIDA supports run-time selection of keyphrases to be considered when computing the contextual similarity. In case an application does not want to completely disregard a keyphrase, but merely decrease its impact on the final decision, U-AIDA additionally supports assigning per-source global weights that are used to alter the importance of a specific keyphrases source.

3.4.5. Entity-Entity Semantic Relatedness

This section addresses the following research issues:

- Different knowledge bases provide different measures of entity-entity relatedness.
- Few corpora contain entity-level annotations.

All modern NERD systems incorporate a notion of entity-entity coherence into their NED techniques. U-AIDA supports two approaches from computing entity-entity semantic coherence:

Co-occurrence-based approach: Entity-entity co-occurrence in the same document signals semantic relatedness between those entities. Co-occurrence statistics are computed from corpora with entity-level annotations. It requires documents where mentions are disambiguated to canonical entities (such as Wikipedia). To this end, U-AIDA takes as an input the entities occurring in each document and computes entity-entity relatedness scores using Milne and Witten [39] formula explained in Section 2.5.

Keyphrases-based approach: In order to overcome the absence of corpora with entity-level annotations, other approaches have been introduced that rely only on purely textual data. KORE [24] leverages overlap in keyphrases to estimate entity-entity relatedness. U-AIDA can be configured to use KORE to compute entity-entity relatedness using the set of keyphrases associated with each entity in the repository.

3.5. Disambiguation Techniques

NERD is never a one-size-fits-all problem. In the same way that different data ingredients may be necessary, different techniques and algorithms can be suitable. For short text that does not contain enough contextual words, coherence is a strong clue. In contrast, for books, for example, employing a coherence measure may significantly penalize the speed without significant improvement in the quality.

U-AIDA is equipped with a suite of disambiguation techniques, and allows adding others when needed. U-AIDA supports disambiguating entities using only prominence scores or considering contextual keyphrases in addition. More specifically, U-AIDA supports two families of disambiguation techniques:

Local Techniques: They solve the NED problem one-mention at a time where each mention is disambiguated independently from other mentions. U-AIDA is equipped with two local techniques. The first considers only entity global prominence. The second is the contextual similarity-based approach introduced in [26].

Joint Techniques: They solve the NED by collectively considering all mentions in the input document. Those approaches require a notion of entity-entity semantic relatedness. U-AIDA is equipped with two joint NED techniques: U-AIDA has an implementation of

the graph algorithm in [26]. In addition, it provides an implementation of the Integer Linear Programming (ILP) approach introduced in [30].

3.6. Summary

In this chapter, we discussed all stages in the NERD pipeline, showing how U-AIDA provides a flexible architecture that supports plugging in different pieces of data as well as incorporating different approaches during the NERD process. In addition, U-AIDA can be configured to combine different data pieces according to the application scenario. This allows each U-AIDA application to build a custom-made NERD solution. In Chapter 4, we will show the power of this flexible architecture by synthesizing NERD systems for two completely different tasks beyond the standard "English News" use case.

4. Applications of the U-AIDA Architecture

In this chapter, we leverage the U-AIDA architecture introduced in Chapter 3 that allows building a NERD system from a mix of general and domain-specific KBs. We show the viability of the architecture by two use-cases. First, we synthesize NERD solution on top of two KBs, YAGO (built from Wikipedia) and GND (covering German authors and literature), and test it on a German corpus obtained from the German National Library. Second, we exploit the U-AIDA architecture to build a system to disambiguate tweets. We participated in the #Microposts Challenge 2014 [61], and the U-AIDA system reached the fourth place out of eight participants [7].

4.1. Domain-Specific Named Entity Disambiguation

4.1.1. Introduction

Depending on the input text, the mentioned entities may belong to multiple domains, and entities are not always covered in Wikipedia. However, they might very well be present in domain-specific KBs. For example, news articles contain prominent entities that can mostly be found in Wikipedia, and hence can be disambiguated against any Wikipedia-based KB. When disambiguating movie reviews, however, IMDb[1] has a higher coverage of entities like movies, actors, and directors. Consider this example text:

> **The Square** is a **Netflix** production documentary movie starring **Ahmed Hassan**, **Khalid Abdalla** and **Magdy Ashour** that depicts **the Egyptian Revolution**

While the entities `The Square` and `Khalid Abdalla` exist in both IMDb and Wikipedia, `Netflix` and `The Egyptian Revolution` exist only in Wikipedia, and only IMDb is aware

[1] `http://www.imdb.com`

of the actors `Ahmed Hassan` and `Magdy Ashour`. Therefore, it is essential to build custom NERD systems where the entities are registered in multiple KBs.

4.1.2. Multi-Knowledge-Base Architecture

Combining Knowledge Bases: In order to disambiguate text that spans multiple domains, and to achieve high coverage, it is essential to combine different KBs to form a single entity repository.

Our architecture deals with this by defining one KB as *reference KB*, and all others are *mapped KBs*. For two entities e_{KB-ref} and e_{KB-map} that belong to the reference and a mapped KB, respectively, where $e_{KB-ref} \equiv e_{KB-map}$, only e_{KB-ref} is added to the entity repository to avoid duplication. The equivalence mapping of entities is assumed to be provided as additional input per KB. Entity equivalence information is provided during the stage of building the NED entity repository as well as other data pieces such as name-entity dictionary or entity keyphrases. U-AIDA expects the mapping data to be provided in the format of Java interface that contains one method:

```
String mapId(String id);
```

Downstream applications are expected to provide their own implementation of that interface.

U-AIDA automatically leverages the mapping information to resolve dictionary entries and keyphrases to the proper canonical entity. For example, given $e_{KB-ref} \equiv e_{KB-map}$ and a dictionary entry "name" $\rightarrow e_{KB-map}$, U-AIDA adds to the name-entity dictionary this entry "name" $\rightarrow e_{KB-ref}$ instead.

Filtering Entities by Type: Often, only parts of a reference or mapped KB should be included. For example, IMDb models actors, directors, and all other persons involved in the movie business with more sophistication than Wikipedia does. Thus, all entities of type `actor`, `director`, etc. should be taken only from IMDb. On the other hand, some domain-specific KBs still have entities pertaining to general world knowledge, e.g. persons or locations, which are usually captured better in Wikipedia.

As explained in Section 3.4, during run time, U-AIDA supports including only parts of KBs based on the semantic types of entites: for example, when building the list of candidate entities, U-AIDA leverages a *Type White List* to take all entities of type `actor` from IMDb and all `organizations` from Wikipedia, thus bringing together the best of both worlds.

Mining Entity Descriptions: Each entity requires a contextual description, e.g., in the form of a set of characteristic keyphrases. These keyphrases are then matched against the context of a mention in the input text to compute a similarity score. Keyphrases can be mined easily from Wikipedia in the form of anchor texts and categories. Domain-specific KBs sometimes lack this kind of description, and almost always come without any semi-structured markup like href anchors or category tags. Still, keyphrases can be mined from some of the relations in the KB. Keyphrases of an `actor` should include the titles of all movies he `actedIn`. Similarly, given a `movie`, names of all actors that `actedIn` in it should be part of its contextual description. In addition, keyphrases can be mined from natural language text such as textual description of entities by extracting all nounphrases or proper nouns. In IMDB, for example, each movie has a plot description that contains entity and characteristic keywords, and hence, can be used to mine keyphrases. In addition, keyphrases can be extracted from general resources such as the social media by extracting nounphrases. U-AIDA combines keyphrases extracted from different sources into the NED entity repository.

Scoring Keyphrases: Keyphrases are assigned scores based on their importance for an entity. Weights can be global inverse document frequencies (IDF) computed from all keywords. If the KB provides a notion of entity occurrence, our architecture leverages it to compute mutual information scores, which are more informative than IDF scores.

Computing Entity Coherence: Coherence is usually computed on the basis of entity co-occurrence in a corpus. Wikipedia readily provides these co-occurrence statistics by its hyperlinks. In domain-specific KBs, though, this is often not present, and coherence needs to be computed differently, e.g. by keyphrase-overlap [24].

4.1.3. The Case of the German National Library

The German National Library (DNB) contains a huge collection of German documents including newspapers, books, and dissertations. They contain many entities that do not exist in the English or the German Wikipedia. The Gemeinsame Normdatei (GND) [2] (Integrated Authority File) is a catalog of entities that is cooperatively supported by many institutions including DNB. The GND contains all German works available in the DNB, together with their authors and publishing organizations. Unsurprisingly, many entities are not prominent enough to appear in the German Wikipedia. However, a NERD solution for disambiguating the documents in the GND should consider all such entities.

[2] `http://www.dnb.de/gnd`

Entities in the GND are organized under seven main classes: `PERSON`, `FAMILY`, `SUBJECT HEADING`, `CORPORATE BODY`, `CONFERENCE OR EVENT`, `WORK`, `PLACE OR GEOGRAPHIC NAME`. Many prominent entities in the GND have equivalent counterparts in the German Wikipedia. Wikipedia is a rich source for extracting different potential names of entities. In addition, Wikipedia helps to build a more comprehensive keyphrases catalog.

Both GND (the more comprehensive entity catalog), and Wikipedia (the rich knowledge resource) should be considered to achieve the best disambiguation quality. In order to disambiguate documents in DNB, we have synthesized a custom-NERD system on top of Wikipedia and GND.

Entity Catalog: Our entity repository picks entities from both the German Wikipedia and the GND. We included locations and organizations from the German Wikipedia because of its high coverage and good level of granularity. All other entities haven been extracted from the GND since it is more comprehensive for other entity types. Specifically, the following entity types have been extracted from the GND: `PERSON`, `FAMILY`, `SUBJECT HEADING`, `CONFERENCE OR EVENT`, and `WORK`.

The entity catalogs of both The GND and the German Wikipedia are not mutually exclusive, many entities are shared among both such as prominent persons. However, we enrich our entity repository with data from both knowledge resources regardless from our entity selection. For example, persons are included only from the GND. However, for persons in the GND that exist in Wikipedia as well, we leverage Wikipedia to add more name-entity dictionary entries as well as keyphrases using the same standard extraction techniques of building AIDA out of Wikipedia.

Entity Name Dictionary: The dictionary is created from entity names in YAGO (extracted from Wikipedia), together with potential names of GND entities. Person names in the GND are extracted from two relations, `preferredNameEntityForThePerson` and `variantNameEntityForThePerson`. For non-persons, names are extracted from relations `preferredName` and `variantName`.

Entity Context: For Wikipedia entities, we follow the same approach of the original AIDA framework by extracting anchor texts, Wikipedia category names, and titles of incoming links as set of characteristic phrases for an entity.

For GND entities, keyphrases are extracted as follows:

1. **The GND:** We extracted keyphrases from different relations in GND. For example, for an author, we harnessed all titles of works written by him. Two relations have

been excluded from keyphrases extraction `oldAuthorityNumber` and `gndIdentifier` because they do not hold adequate keyphrases.

2. **Title Data:** In addition, keyphrases have been extracted from tables of contents as well as abstracts of different works in the GND and associate them with the respective authors. We applied regular expression on part-of-speech patterns to extract proper noun phrases. The patterns are taken from [41].

e-e Coherence: Semantic-relatedness between entities has been estimated using overlap between keyphrases. We applied the KORE technique introduced in [24].

4.1.4. Experiments and Evaluation

Setup: We synthesized a U-AIDA solution out of around 200K entities from the German Wikipedia and around 10M entities from GND including the explained entity filter. We randomly sampled paragraphs out of the electronic documents submitted to the DNB in 2014. Documents have been manually annotated by two fluent German-speaking students. Conflicts have been resolved by another native German speaker. All in all, this test corpus has 171 documents having 1572 mentions that span 13 main domains.

Results and Discussion: We computed the confidence score as introduced in [23]. Similarity scores are normalized on a per mention basis as follows:

$$\text{norm}_{score}(m, e) = \frac{\text{score}(m, e)}{\sum_{e_i \in E_m} \text{score}(m, e_i)}$$

where E_m is the set of candidate entities for a mention m.
The confidence of correctly mapping a mention m is computed as follows:

$$\text{conf}_{norm}(m) = \text{norm}_{score}(m, \arg\max_{e \in E_m} \text{score}(m, e))$$

We threshold the mapping on different levels of confidence. All mappings of lower confidence scores are ignored. We evaluated the system at different confidence levels using micro precision and recall. Recall reflects the number of mentions that have been mapped with confidence score higher than the confidence threshold. The precision represents the percentage of correct mappings out of the mappings with confidence higher than the threshold.

The micro precision and recall are given at different levels of confidence as shown in Table 4.1. The overall performance is worse than news articles, the typical benchmark for

	Precision	Recall
NED	58.86	100
NED (Conf >0.2)	65.14	53.33
NED (Conf >0.5)	70.85	44.27

Table 4.1.: Results of Running U-AIDA on a Sample Corpus of DNB Documents

NED. This is due to the fact that the entity repository of the GND is much larger and thus more ambiguous than Wikipedia, but at the same time provides less features to be used for the actual disambiguation. However, the experiments show the applicability of the architecture on combining completely different KBs.

4.2. Named-Entity Disambiguation for the Social Stream

4.2.1. Introduction

Microblogs present a rich field for harvesting knowledge, especially Twitter with more than 500 million tweets per day [27]. However, extracting information from short informal microposts (tweets) is a difficult task due to insufficient contextual evidence, typos, cryptic abbreviations, and grammatical errors. By exploiting the U-AIDA architecture, we managed to synthesize a robust NERD solution to handle short microposts by adding additional components for named entity recognition, name normalization, and extended candidate entity retrieval. We also integrate data harvested from the Twitter API into our model to cope with context sparsity. Moreover, we tuned the AIDA parameters to accommodate the brief informal nature of tweets.

4.2.2. Adapting U-AIDA to Tweets

Named Entity Recognition: AIDA originally uses the Stanford Named Entity Recognition (NER) tagger, with a model trained on newswire snippets, a perfect fit for news texts. However, it is not optimized for handling user generated content with typos and abbreviations. Hence, we employ two different components for mention detection: The first is Stanford NER with models trained for case-less mention detection; the second is a dictionary-based NER tool [5]. The dictionary-based NER is performed in two stages:

1. **Detection** of named entity candidates using dictionaries of all names of all entities in our knowledge base, using partial prefix-matches for lookups to allow for shortening of names or little differences in the later part of a name. For example, we would

recognize the ticker symbol "GOOG" even though our dictionary only contains "Google". The character-wise matching of all names of entities in our KB is efficiently implemented using a prefix-tree data structure.

2. **Filtering** the large number of false positives using a collection of heuristics, e. g. the phrase has to contain a NNP tag or it has to end with a suffix such as "Ave" in "Fifth Ave".

Mention Representation: We exploit the flexible input representation of the U-AIDA architecture to automatically assign potential name variants to each mention. For example, the hashtag "`#BarackObama`" should be normalized to "Barack Obama" before matching it against the dictionary. Furthermore, many mentions of named entities are referred to in tweets by their Twitter user ID, such as **`''@EmWatson''`**, the Twitter account of the British actress "Emma Watson". Because the Twitter user IDs are not always informative, we access the account metadata, which contains the full user name.

We attach to each mention string a set of potential mention representations and use all of them to query the dictionary. For example, **`''@EmWatson''`** will have the following potential representations:

1. **"EmWatson"**: The original mention string without the "@" character.
2. **"Em Watson"**: The original mention string after adding a blank space before every uppercase characters.
3. **"Emma Watson"**: The name in the Twitter profile data. It is automatically obtained using the Twitter API.

Therefore, this is the set of the potential mention representations {"EmWatson", "Em Watson", "Emma Watson"}, and each of them will be matched against the dictionary to retrieve the set of candidate entities.

The original AIDA system incorporates prior probability as a measure for entity prominence. Prior probability is a function of both the mention and the entity. Since each mention m_i is now a set of potential representations $N(m_i)$, we adapt the prior probability measure as follows. The prior probability of an entity e_i given a mention m_i:

$$prior(m_i, e_i) = \max_{m' \in N(m_i)} prior(m', e_i)$$

where $N(m_i)$ is the set of potential mention representation of m_i. The maximum is taken in order not to penalize an entity if one of the normalized mentions is rarely used to refer to it.

Approximate Matching: This step is employed if and only if the previous normalization step does not produce candidate entities for a given mention. For example, it is not obvious how to automatically split a hashtag like ‘‘`#londonriots`’’, and hence its potential representations set is {“londonriots”}. Querying the name-entity dictionary with this name “londonriots” does not retrieve any candidate entities.

We address this by representing both the mention strings and dictionary keys as vectors of character-trigrams. We use trigram matching to assess the similarity between a mention string and a name in the name-entity dictionary. For example, considering the mention “londonriots”, it is represented as the vector:

⟨lon,ond,ndo,don,onr,nri,rio,iot,ots⟩

Similarly, a dictionary key “London Riots” will be represented as the vector:

⟨Lon,ond,ndo,don,on ,n R, Ri,Rio,iot,ots⟩

We compute the cosine similarity between both vectors (in this case 0.66667). If the similarity score is above a certain threshold (experimentally determined as 0.6), we consider the mention string and the dictionary key similar, and we assign all candidate entities of the dictionary key to the mention string.

Computing cosine similarity between vectors of character trigrams is computationally expensive. Therefore, we only use this as a fallback option when the mention has no candidate entities assigned to it. Furthermore, we use the Postgres pg_trgm module [3] that creates the appropriate indexes for fast searching for similar strings. We use the Postgres implementation for computing the similarity. It creates the trigrams in a slightly different way: it considers the string to be prefixed by two spaces and suffixed by one space when determining the set of trigrams. For example, the set of trigrams in the string “riots” is

⟨ r, ri,rio,iot,ots,ts ⟩

[3] `http://www.postgresql.org/docs/9.1/static/pgtrgm.html`

Unlinkable Mentions: Some mentions should not be disambiguated to an entity, even though there are candidates for it. This is especially frequent in the case of social media, where a large number of user names are ambiguous but do not refer to any existing KB entity – imagine how many Will Smiths exist besides the famous actor. We address this problem by thresholding on the disambiguation confidence as defined in [23], where a mention is considered unlinkable and thus removed if the confidence is below a certain threshold (experimentally estimated as 0.4).

4.2.3. Experiments

Data

We conducted experiments on the dataset provided in the #Microposts2014 Challenge [7]. The data comprises 3,505 tweets extracted from a collection of over 18 million tweets. The tweets are extracted in the period from 15th July 2011 to 15th August 2011 (31 days). They cover events such as the death of `Amy Winehouse` and the `London Riots`; these are likely to include named entities.

The challenge evaluates both extraction and disambiguation of entities. 70% of the tweets are used for training and released to the participants together with the ground truth entities. The training set contains 2,340 tweets, with 41,037 tokens and 3,819 mentions of named entities. The test set contains 1,165 tweets, with 20,224 tokens and 1,458 mentions of named entities.

Setup

We carried out experiments with three different setups:

1. First, we used Stanford NER trained for entity detection, along with mention prior probability and key-phrases matching for entity disambiguation.

2. In the second experiment, we added coherence graph disambiguation to the previous setting.

3. The third setting is similar to the first one, but we use the dictionary-based NER instead of Stanford NER for entity detection.

Since the data has mentions of numbers annotated, we automatically annotate all digit-only tokens as mentions using a regular expression.

Parameter Settings

In the original AIDA framework, the weight of a mention-entity edge is computed by a linear combination of different similarity measures. In order to estimate the coefficients of the linear combination, we further split the released tweets training dataset into `TRAIN` and `DEVELOP` chunks. We used the `TRAIN` chunk for estimating the coefficients of the linear combination. We estimated further hyper-parameters for our algorithm (like the importance of mention-entity vs. entity-entity edges) on `DEVELOP`.

Results and Discussion

On the `DEVELOP` part of the training data, our experiments achieved around 51% F1. We submitted three runs on the `TEST` data. The challenge moderators reported only the result for the best run which in our case is the third variant which utilizes dictionary-based NER and incorporates coherence measures. The precision, recall and F1 scores of the U-AIDA system on the `TEST` data are 53.28%, 39.51%, and 45.37% respectively.

The scores are relatively low because NERD problem on tweets is significantly harder than on news. In addition, the evaluation is strict in the sense that a mention is counted as true positive only if both the mention span matches the ground truth perfectly and the entity label is correct.

5. U-AIDA for Languages with Poor Annotated Resources

In this chapter, we address the NED problem for languages with limited amounts of annotated corpora and entity-structured resources such as Arabic. We present a method that leverages structured English resources to enrich the components of the language-agnostic U-AIDA framework and enable effective NED for other languages. We achieve this by fusing data from several multilingual resources and the output of automatic translation/transliteration systems. We show the viability and quality of our approach by synthesizing NED systems for Arabic, Spanish and Italian.

5.1. Overview

While the English Wikipedia is relatively large and suitable for producing sufficiently large dictionaries and entity descriptions, this is not the case for other languages such as Arabic which has an order of magnitude smaller Wikipedia. This dramatically affects the size of the aforementioned resources necessary for the NED task. For example, compiling an Arabic dictionary of entity names only from the Arabic Wikipedia produces a small dictionary that misses names of many prominent entities.

Nevertheless, such scarcity of semi-structured Arabic data is accompanied with huge growth in the online Arabic content such as blogs and news articles. Arabic is among the most widely spoken languages on the Internet. However, it is a resource-poor language in the sense that there exist only small annotated corpora and entity-structured resources.

Our approach to overcome these bottlenecks is based on enriching the following NED building blocks:

1. Entity Catalog.
2. Name-Entity Dictionary.
3. Entity Keyphrases Catalog.

The following sections discuss each of the three buildings blocks and how they can be enriched to improve the performance of NED on languages with limited resources.

λ	A language in Wikipedia
L	Set of all languages in Wikipedia
e_{en}	An entity originated from the English Wikipedia
e_{λ}	An entity originated from the λ Wikipedia
e	An entity in the final entity repository
E	Set of all entities
$Cat_{en}(e)$	Set of categories names of an entity e in the English Wikipedia
$Cat_{\lambda}(e)$	Set of categories names of an entity e in the λ Wikipedia
$Inlink_{en}(e)$	Set of Incoming Links to an entity e in the English Wikipedia
$Inlink_{\lambda}(e)$	Set of Incoming Links to an entity e in the λ Wikipedia
$Trans_{en \to \lambda}(S)$	Translation of each element in S from English to λ

Table 5.1.: Terminology Used for Building an NED System for a Language λ

Terminology used throughout this chapter is introduced in Table 5.1. Although this chapter discusses Arabic as an example, our techniques are language-agnostic in the sense that they can be applied to any other language as shown in the experiments section.

Let us consider this hypothetical *Arabic* sentence:

قد يتم ترشيح **عائض القرني** لنيل **جائزة غوته** عن كتابه **لا تحزن**

and written in English for clarity as:

Aaidh Al-Qarni might get nominated for
Goethe Prize for his book **La Tahzan**

This sentence has three named entities: the writer `Aaidh Al-Qarni`, the prize `Goethe Prize` and the book `La Tahzan`. We will show how we can adapt different data components of an NED framework to be able to correctly disambiguate all of them.

5.2. Entity Catalog

The writer entity in the example is famous enough to exist in both the English and Arabic Wikipedias. Nevertheless, the book itself exists only in the Arabic Wikipedia. On the other hand, the prize is not known to the Arabic Wikipedia and exists only in the English Wikipedia (as of 01 May 2015).

In order to build a system capable of disambiguating this sentence, we need to make sure our entity repository contains all of those entities. This is done by considering inter-wiki links. If an entity in language $\lambda \in L - \{en\}$ has an English counter part, the English one is kept instead of that in language λ, otherwise, the original entity is kept. For example, in our repository, the entity used to represent Egypt is "Egypt" coming from the English Wikipedia instead of "ar/مصر" coming from the Arabic Wikpedia. However, the entity that refers to the western part of Cairo is identified as "ar/غرب القاهرة" because it has no counterpart in the English Wikipedia.

Considering our example, the following entries are added to the entity repository.

Available Entities		Entity ID in the final repository
`Aaidh_Al-Qarni`, ar/عائض القرني	→	`Aaidh_Al-Qarni`
`Goethe_Prize`	→	`Goethe_Prize`
ar/لا تحزن	→	ar/لا تحزن

In general, for building an NED system for a language λ, we combine the Wikipedia entities for *en* and λ. This captures prominent entities from the English side and local entities that are peculiar only to this language or culture from the λ Wikipedia. We used YAGO3[35] as our back-end KB which is constructed from Wikipedia entities of different languages.

5.3. Name-Entity Dictionary

Considering the Arabic example above, in order to disambiguate it correctly we need a dictionary that is aware of the Arabic names of all the three entities. Our system knows, at least, one Arabic name only for entities that exist in the Arabic Wikipedia (their Wikipedia page titles).

For constructing the dictionary, we harness Wikipedia page titles, disambiguation pages, redirects, and anchor text. Since we are building an Arabic dictionary of entity names, we restricted our extraction to the Arabic Wikipedia. Due to the small size of the Arabic

Wikipedia, the dictionary extracted from it is very incomplete. Not only does it miss Arabic names for prominent entities (which do not exist in the Arabic Wikipedia) but it additionally misses important Arabic name variations for other entities. Therefore, it is crucial to enrich and extend the dictionary by adding entries other than those extracted from the Wikipedia.

We have developed three approaches to extend the name-entity dictionary of our system as explained below.

5.3.1. External Resources

The entities that exist only in the English Wikipedia lack Arabic names, but their English name is available. The most straightforward approach is to leverage name dictionaries so-called *gazetteers*, that are designed to translate named entities (as opposed to concepts) from English to Arabic. More specifically, we aim at including potential entity names mentioned in unstructured web resources. Therefore, we harvested entity mentions from two kinds of external resources.

Entity-aware dictionaries: We harness names linked to entities through manually created anchors linking to Wikipedia pages. Spitkovsky and Chang, 2012 [47] created a multilingual bi-directional dictionary from strings, including non-Wikipedia web anchors, to Wikipedia articles. Each mapping between the strings and the associated article has conditional probability $P(\text{URL}|\text{string})$ defined as the ratio of the number of hyperlinks into a Wikipedia URL having anchor text "string" and the total number of anchors with text "string".

Strings with conditional probability less than a threshold (experimentally picked to 0.01) were filtered out. This excludes the common non-expressive strings such as *"read more"* and *"Wikipedia"*. We used the language detection module developed by [46] to extract strings of the target language. A further token-level cleaning is applied to remove URL strings and to remove stop words, punctuation and other common marker prefixes or suffixes from the strings such as *"Wikipedia page"* in *"Germany Wikipedia page"*. Finally, names are associated to entities through Wikipedia URLs.

Entity-free dictionaries: There are several multilingual name dictionaries (gazetteers) that are manually (or semi-manually) populated from parallel or comparable corpora. Names in these resources are not mapped to specific entities. Therefore, the targeted language mentions are associated with all entities that have a matching English name. More specifically, given a name-entity dictionary entry ($\text{name}_{en} \Rightarrow e$) denoting that

"name$_{en}$" is a potential English name for the entity e, and a multilingual dictionary entry (name$_\lambda \equiv$ name$_{en}$), we add (name$_\lambda \Rightarrow e$) to the name-entity dictionary.

We used JRC-Names [48] as a lookup dictionary to generate variants for the English names in the target language. In addition, we leveraged the parallel entity names harvested by Azab et al. 2013 [4] from parallel news wires corpora using a semi-automated approach.

5.3.2. Statistical Machine Translation

While gazetteers and hyperlinks extracted from the web provide Arabic names for some English entities, many are still nameless in the Arabic world. There is a need to *generate* Arabic names instead of only extracting them.

Several off-the-shelf translation services have been trained with huge amounts of data (e.g. Google Translate and Microsoft Translator). However, these services are not geared for translating entity names. Instead, they are designed to achieve good translation quality on natural language text where "Green" is mostly a *color* and "North" is a *geographic direction*. Neither of them would be considered as a *family name*. More specifically, unless the name to translate has appeared in the English-Arabic parallel training data (which is seldom the case for not so prominent entities), the translation quality is not satisfactory [1, 22, 4].

We trained an SMT system on carefully selected parallel corpora of names. The intuitive idea of our approach is the following. If our KB knows the names of "Eric **Schmidt**" and "**Christian** Dior" in Arabic script, our system should be able to automatically learn the Arabic name of "**Christian Schmidt**".

SMT System: We used cdec [12]: a full fledged SMT system. We configured cdec to use the default word aligner and model optimizer.

Training Data: We trained an SMT system on an English-Arabic parallel corpus of names. We obtained the corpus from the following resources:

1. **External Resource**: We included the dictionary provided by [4].

2. **Wikipedia**: We automatically built a parallel corpus using interwiki links. For every entity that exists in both the English and Arabic Wikipedia, its English and Arabic names are added to the training corpus. Obviously, this approach is not limited to Arabic and can be applied to any other language.

5% of the training data were used for tuning the parameters of the cdec decoder.

Approach: Recent techniques for translating named entities within natural language context trained a classifier on contextual and linguistic features including the coarse-grained type tags (`PERSON`, `LOCATION`, `ORGANIZATION`) [4]. We also took *type information* into consideration when translating names as explained below.

Every name pair in our training data has type information. The dictionary obtained from [4] classified names into standard coarse-grained classes (`PERSON`, `LOCATION`, `ORGANIZATION`, `MISC`). The training data obtained from Wikipedia are extracted from entity names, and hence their semantic types are known via their entity semantic types.

Training data have been split into two sets: `PERSONS` that contains the names typed as persons, and `NON-PERSONS` that contains the rest of the training data. We have trained three SMT systems:

1. `PERSON-SMT`: It is an SMT system trained only on the names in the training data typed as `PERSON`.

2. `NON-PERSON-SMT`: It is an SMT system trained only on the names in the training data typed as `NON-PERSON`.

3. `BOTH-SMT`: It is an SMT system trained on all names in the training data.

Given an entity of type `PERSON`, we translate its English name using the system `PERSON-SMT`. Only if no translation is not found, we invoke the `BOTH-SMT` system. Similarly, a `NON-PERSON` entity name, is first translated using the `NON-PERSON-SMT` system. If it fails, it is translated using the `BOTH` system. The `BOTH-SMT` is only used as fallback system.

The intuitive idea behind this approach is that names of different types are treated differently. Consider, for ease of explanation, translating names from English to German. The word "Green", for example, has two different German translations in the names "John Green" and "East German Green Party". The former is a person, and "Green" would stay as is. The latter is a political party, and "Green" will be translated to "Grün". Training the SMT system on per-type data sets allows it to learn the most common translation for "Green", for example, in each type. We did not do any further type-based data splitting (e.g. `ORGANIZATION` vs `LOCATION`) in order to maintain a reasonable amount of training data.

If the type-specific SMT system is not able to translate the name, we switch to the fallback system trained on all training data. The intuition here is that for non-person entities like "Goethe Prize", we are able to translate it using the fallback system which learned to translate "Goethe" from the `PERSON` part of the training data, and "Prize" from the `NON-PERSON` part.

5.3.3. People Name Transliteration

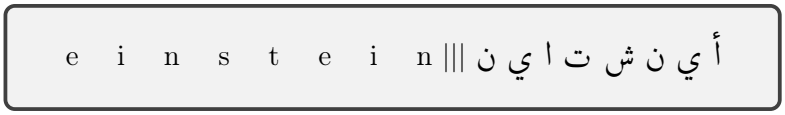

Figure 5.1.: Character-level Training Data Point Example

Many entities can be assigned one or more Arabic names using external resources and/or SMT. However, some entities are still left without any names. Furthermore, due to different dialects and non-unique transliteration variants, some potential entity names are missing. For example, the entity `Angela Merkel` has only "أنجيلا ميركل" as an Arabic name. Other variations such as "أنغيلا ميركل", which may appear in news articles, should be automatically generated. Transliteration is the key to be able to generate other forms of names. We use cdec SMT system trained on the same parallel corpus of English-Arabic name pairs that was used for translation. However, we train it on the character level in order that it learns the potential equivalent Arabic letter for each English letter. An example is shown in Figure 5.1.

In order to avoid adding noisy entries to the dictionary, we apply transliteration only for entities of type `PERSON`. Applying transliteration for entities of type `ORGANIZATION` for example will produce wrong entries by transliterating entities such as `United_Nations` instead of translating them. The reason is that most of organization names need partial or full translation (as opposed to transliteration) when generating names in other languages.

5.4. Entity Descriptions

NED requires textual description of each entity in language λ that gets matched against the input text to compute a similarity score between this entity and the input text. We adapted the standard approaches to extract entity contexts as follows:

- **Anchor Text**: We extract all anchor texts in an entity page in the λ Wikipedia under the assumption that all anchor texts in a λ Wikipedia are of language λ. Nevertheless, for languages with small Wikipedia such as Arabic, many entities are missing. Those missing entities do not receive any Arabic keyphrases from this source.

- **Inlink Titles**: In AIDA, the set of the titles of the pages that has links to an entity were considered among the keyphrases of such an entity. We pursue the same approach here, and *fuse* incoming links to an entity from both the English and the λ Wikipedias. Then, we translate the titles that are not of language λ using interwiki links. For those entities that do not have a counter part in the λ Wikipeida, their names cannot be translated using interwiki link. Therefore, we invoke an SMT system trained on a parallel corpus of entity names built from interwiki links, to translate them. Formally, the set of keyphrases extracted from incoming links are as follows:

$$Inlink(e) = Inlink_{\lambda}(e) \cup \underset{en \rightarrow \lambda}{\text{Trans}}(Inlink_{en}(e))$$

- **Category Names**: For an entity, we include the category names in both the English and the λ Wikipedias. We exploit the interwiki links among categories to translate the English categories into language λ. This comes with two benefits. First, we use the category mappings which result in fairly accurate translation in contrast to machine translation. Second, we enrich the category system of the λ Wikipedia with categories from English for entities that have corresponding English counterparts. Similar to incoming links, for those category names that cannot be translated using interwiki links, we use an SMT to translate them. However, the SMT system is trained on a parallel corpus of category names built from interwiki links. Formally, the set of keyphrases extracted from category names are computed as follows:

$$Cat(e) = Cat_{\lambda}(e) \cup \underset{en \rightarrow \lambda}{\text{Trans}}(Cat_{en}(e))$$

Table 5.2 summarizes which context resource has been translated and/or enriched from the English Wikipedia.

The aforementioned approaches leverage cross-lingual evidences, such as category names, to enrich the set of entity keyphrases in a language with relatively small Wikipedia. This enables our system to deliver a NERD solution for language λ with name-entity dictionary and entity keyphrases with coverage close to the English counterpart. We maintain quality by relying first on interwiki links that deliver more accurate translation than that obtained by SMT systems.

Context Source	λ Wikipedia	English Wikipedia
Anchor Text	+	-
Categories	+	+
Title of Incoming Links	+	+

Table 5.2.: Entities Context Sources when Building an NED System for Language "λ"

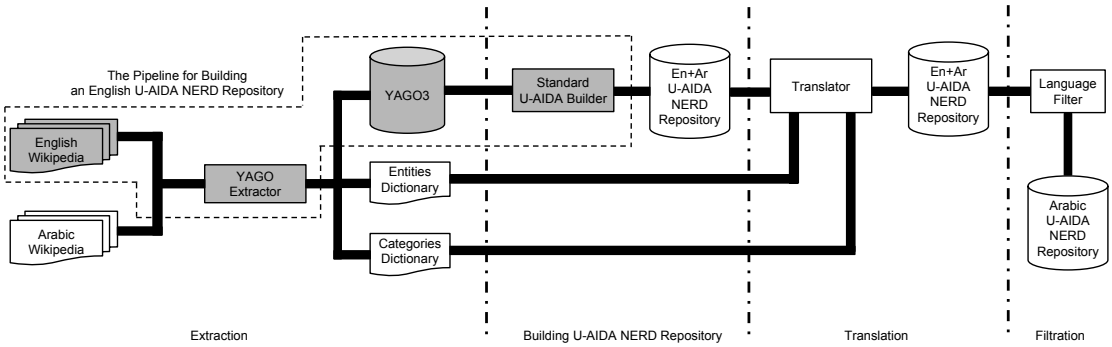

Figure 5.2.: General Architecture for Building an NED System for Arabic

5.5. Implementation

Figure 5.2 illustrates the general architecture of building an NED solution for the Arabic language. The pipeline is divided into four stages:

1. **Extraction:** In this stage, the entity catalog is built from the English and Arabic Wikipedias taking equivalent entities into account. In addition, dictionaries for entity names and category names are extracted from the interwiki links.

2. **Building the NERD Repository:** In this stage, all data ingredients required for NERD, such as anchor texts, page titles, disambiguation pages, are extracted from both Wikipedias. The output is a NERD repository that contains a mix of English and Arabic entity names and keyphrases.

3. **Translation:** Data extracted in the second stage are then translated using interwiki links, external dictionaries, and SMT systems. In addition, person names get transliterated.

4. **Filtration:** In this stage, data are cleaned to remove any text that could not be translated in the previous stage. The output is a purely Arabic NERD repository.

5.6. Experiments and Evaluation

Setup

We evaluated the quality of our methods for enriching the NED repository by running experiments with and without the enrichment. We tested our approach on Arabic, Spanish and Italian languages. We built U-AIDA systems referred to as **U-AIDA-Arabic**, **U-AIDA-Spanish**, and **U-AIDA-Italian**. We used YAGO3 as our backend KB built from the English Wikipedia dump as of 12 January 2015 combined with language specific Wikipedias. For Arabic, Spanish and Italian, we used the latest dumps available for each language, namely, 18 December 2014, 08 April 2015, and 25 April 2015, respectively.

The same configuration was used as in the original AIDA local similarity technique [26]. Hyperlinks among entities exhibit different densities in different Wikipedias. Since e-e coherence is estimated using overlap in incoming links, different densities produce coherence scores that are incomparable. Therefore, we did not include coherence measures in our experiments to be able to precisely assess the impact of data enrichment.

Language	Dataset	Type	#docs	# non-null mentions	# unique non-null mentions
Arabic	LDC2014T05	news	702	14413	3109
	LDC2014T05	web	74	1385	514
Spanish	SemEval 2015	web	4	29	21
	ProjSynd	news	683	743	427
Italian	SemEval 2015	web	4	33	22
	Europarl v.5	news	922	1051	434

Table 5.3.: Data Sets Used to Evaluate U-AIDA++ per Language

Datasets

For Arabic, we created our own benchmark to overcome the lack of annotated data. We used *LDC2014T05*[33] corpus with news articles and web pages in English and Arabic where the latter was manually translated and word-aligned. We automatically created pseudo ground-truth by applying the original AIDA system optimized for English on the English text and projecting the annotations onto the Arabic corpus using the token level alignment information. We manually sampled 106 mention annotations. 22 mentions were mapped to null. Out of the other 84 mentions that were assigned entities, 70 were correctly disambiguated for a mapping precision of 83.3%.

In addition, we ran tests with the data of the Spanish and Italian *SemEval-2015* Shared Task 13. Since the task is designed for both Word Sense Disambiguation and Named Entity Disambiguation, we tested only for the latter part. Finally, we evaluated our methods on the Spanish *ProjSynd* and the Italian *Europarl* news commentary data. In both data sets, person mentions have been manually annotated by [36]. We excluded all documents without mentions annotation from our experiments. Table 5.3 summarizes the properties of the data sets used in this evaluation.

Baselines

We compared three systems:

1. **U-AIDA-λ**: language-specific instance of U-AIDA *without* data enrichment.

2. **U-AIDA-λ++**: language-specific instance of U-AIDA *with* data enrichment.

3. **Babelfy**: via a web service.

We built systems for Arabic, Spanish and Italian, referred to as **U-AIDA-Arabic**, **U-AIDA-Spanish**, and **U-AIDA-Italian** that only contain data from Wikipedia without any data extension approaches. We built similar systems (referred to as **U-AIDA-Arabic**++, **U-AIDA-Spanish**++ and **U-AIDA-Italian**++) with the entity names dictionary and entity keyphrases extended with additional entries using the data enrichment approaches explained above.

For **Babelfy**, we used the web service. It offers two modes: named entities full matching and partial matching. We ran both using a predefined set of mentions and report the better results. We limited the candidate space to Wikipedia named entities.

Evaluation

For the Arabic benchmark, we considered only mentions with non-null pseudo ground truth annotations for evaluations. For fair comparison, annotations returned as null by all systems were considered to be wrong. We computed both per-mention average precision and per-document average precision. Precision is computed as the ratio of the number of correct annotations and the number of all annotations returned by the respective system. Results of our experiments are shown in Table 5.4.

Discussion

U-AIDA-λ++ delivered better precision than all other systems under test. The scores are generally higher for Spanish and Italian than for Arabic, because of their more comprehensive Wikipedias. Babelfy does not implement entity name translation [43] which explains its poor performance on the Arabic data. The impact of NED repository enrichment is less expressed for the news corpus compared to the web corpus because many entities in the news corpus are prominent and already exist in the Arabic Wikipedia. Due to technical issues of the Babelfy web service, we could not report its scores on the Europarl data set. Furthermore, U-AIDA-λ++ consistently managed to map, at least, the same number of mentions that U-AIDA-λ could map across different data sets, indicating that data enrichment improved both precision and recall.

In order to assess the impact of different system components, we sampled the Arabic results for mentions that were correctly disambiguated only after applying the enrichment techniques. Arabic is the most challenging among our test languages and has the smallest Wikipedia. The outcome is quite promising. For example, names "اتفاقية كوتونو" and

"ساوث فلوريدا صن سنتينيل" were correctly linked to "Cotonou Agreement", and "Sun-Sentinel newspaper" respectively, although neither has an Arabic Wikipedia page. Some English names have different potential forms in Arabic. In such cases, transliteration produced all possible forms. For example, the Arabic Wikipedia page of the Nobel prize winner "José Saramago" lists his name as "جوزيه ساراماغو". However, in our news corpus, the name "خوسيه ساراماغو" was used instead. Our system could acquire both forms and correctly disambiguate this mention.

	Dataset	System	Mention Precision	Document Precision	Non-null Entities
Arabic	LDC news	U-AIDA-Arabic++	**73.23**	**71.34**	**94.69**
		U-AIDA-Arabic	69.07	67.26	87.19
		Babelfy	30.32	31.16	39.75
	LDC web	U-AIDA-Arabic++	**68.16**	**60.10**	**93.86**
		U-AIDA-Arabic	62.02	52.48	85.56
		Babelfy	22.33	21.13	38.62
Spanish	SemEval	U-AIDA-Spanish++	**86.20**	**69.60**	**93.10**
		U-AIDA-Spanish	82.76	68.04	**93.10**
		Babelfy	79.31	66.47	79.31
	ProjSynd	U-AIDA-Spanish++	**86.91**	**88.04**	**89.98**
		U-AIDA-Spanish	84.71	86.27	87.90
		Babelfy	79.30	86.27	79.94
Italian	SemEval	U-AIDA-Italian++	**87.88**	**88.39**	**87.87**
		U-AIDA-Italian	84.84	86.60	**87.87**
		Babelfy	75.75	49.11	78.78
	Europarl	U-AIDA-Italian++	**75.38**	**75.29**	**85.53**
		U-AIDA-Italian	71.62	71.77	81.20
		Babelfy	n.a	n.a	n.a

Table 5.4.: Experimental Results of Running U-AIDA on Arabic, Spanish and Italian

6. HYENA: Named Entity Type Classifier

U-AIDA architecture enables building multi-knowledge base NERD solutions. However, Named-Entity Disambiguation (NED) assigns some mentions to null indicating that the entities denoted by those mentions do not exist in any of the underlying knowledge bases. The reason can be an incomplete knowledge base or a newly emerging entity. In such cases, U-AIDA can infer the lexical type labels for those mentions using the HYENA system.

Inferring the lexical type labels is an important asset for NLP tasks like semantic role labeling and named entity disambiguation. Prior work has focused on flat and relatively small type systems where most entities belong to exactly one type. In this chapter, we address the more demanding case of fine-grained types organized in a hierarchical taxonomy, with several hundreds of types at different levels. We present the HYENA method that uses a multi-label hierarchical classifier to solve this problem. HYENA exploits gazetteer features and accounts for the joint evidence for types at different levels. Experimental comparisons to the best prior methods and an extrinsic study on named entity disambiguation demonstrate the practical viability of the HYENA method.

6.1. Introduction

Motivation: Web contents such as news, blogs and other social media are full of named entities. Recognizing them and disambiguating them has been intensively studied (see, e.g., [18, 9, 39, 26, 45]). Each entity belongs to one or more *lexical types* associated with it. For instance, an entity such as *Bob Dylan* should be assigned labels of type `Singer`, `Musician`, `Poet`, etc., and also the corresponding supertype(s) (hypernyms) in a type hierarchy, in this case `Person`. Such fine-grained typing of entities and entity mentions in texts can be a great asset for various NLP tasks including semantic role labeling and sense disambiguation. Most notably, Named Entity Disambiguation (NED) can be boosted by knowing or inferring a mention's lexical types. For example, noun phrases such as "songwriter Dylan", "Google founder Page", or "rock legend Page" can be easily mapped

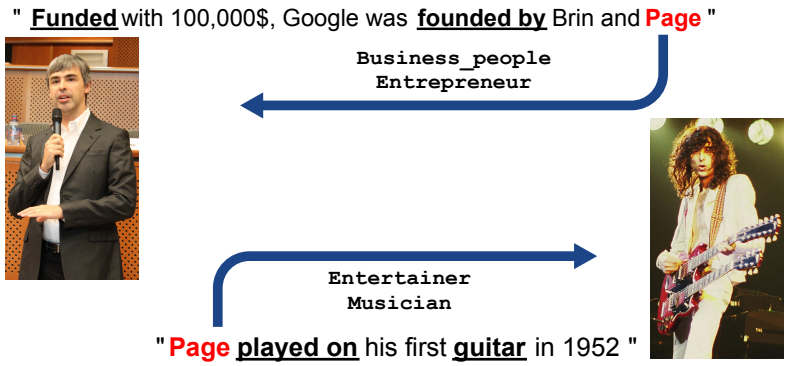

Figure 6.1.: Fine-grained entity type classification

to the entities Bob Dylan, Larry Page, and Jimmy Page if their respective types `Singer`, `BusinessPerson`, and `Guitarist` are available. Figure 6.1 shows an illustrative example.

Problem Statement: State-of-the-art tools for Named Entity Recognition (NER) systems like the Stanford NER Tagger [18] compute such lexical tags only for a small set of coarse-grained types: `Person`, `Location`, and `Organization` (plus tags for non-entity phrases of type time, money, percent, and date). There is little literature on *fine-grained typing* of entity mentions [19, 13, 52, 34], and these approaches are pretty much limited to flat sets of several dozens of types. Because of the relatively small number of types, an entity or mention is typically mapped to one type only. The goal that we address is to extend such methods by automatically computing lexical types for entity mentions, using a large set of types from a hierarchical taxonomy with multiple levels. In this setting, many entities naturally belong to multiple types. For example, a guitarist is also a musician and a person, but may also be a singer, an actor, or even a politician as well. So we face a *hierarchical multi-label classification* problem [51].

Contribution: We introduce the *HYENA* method (Hierarchical tYpe classification for Entity NAmes). HYENA is a multi-label classifier for entity types based on hierarchical taxonomies derived from WordNet [16] or knowledge bases like YAGO [49] or Freebase [6]. Our approach uses a suite of features for a given entity mention: neighboring words and bigrams, part-of-speech tags, and also phrases from a large gazetteer derived from state-of-the-art knowledge bases. Moreover, we carefully consider the hierarchical structure of the type space by exploiting the relatively large amount of training instance in the top-level types, in order to filter our clearly wrong instances. Based on our classifiers, we

also extend a state-of-the-art joint-inference method for entity disambiguation and show that type labels for entity mentions can improve NED performance.

Our salient contributions are the following:

- The first method for entity-mention type classification that can handle multi-level type hierarchies with hundreds of types and multiple labels per mention.
- Extensions to consider cross-evidence and constraints between different types, by developing a meta-classifier.
- Experimental comparisons to three state-of-the-art baselines, demonstrating the superiority of HYENA.
- An extrinsic study on boosting NED by harnessing type information.

The rest of this chapter is organized as follows. Section 6.2 presents our computational model and feature sets. Section 6.3 introduces our hierarchical multi-label classifier, and its extensions into a meta-classifier. Section 6.4 discusses our experiments on type tagging. Section 6.5 presents an extrinsic study on NED. Section 2.4 reviews related work. Section 6.6 presents implementation details.

6.2. Computational Model and Feature Set

We apply a supervised machine learning approach to train a set of classifiers for labeling entity mentions (i.e., noun phrases denoting entities) with lexical types. This section describes the type system and the text features we are using for classification. Section 6.3 will explain the algorithmic aspects of the classifiers.

6.2.1. Fine-grained Type Hierarchy

We have systematically derived a very-fine-grained type taxonomy from the YAGO knowledge base [49, 25] which comes with a highly accurate mapping of Wikipedia categories to WordNet synsets. We start with five broad classes namely `PERSON`, `LOCATION`, `ORGANIZATION`, `EVENT` and `ARTIFACT`. Under each of these superclasses, we pick 100 prominent subclasses. The selection of subclasses is based on the population of the classes: we rank them in descending order of the number of YAGO entities that belong to a class, and pick the top 100 for each of the top-level superclasses. This results in a very-fine-grained reference taxonomy 505 types, organized into a directed acyclic graph with 9 levels in its deepest parts. Table 6.1 lists the top 20 subtypes under each of the 5 top-level types. The full type hierarchy is listed in Appendix A.

PERSON	LCOATION	ORGANIZATION	EVENT	ARTIFACT
contestant	location	institution	social event	instrumentality
athlete	region	unit	act	medium
player	geographical area	company	show	structure
leader	district	educational institution	movie	creation
entertainer	administrative district	school	activity	product
intellectual	tract	association	contest	album
performer	site	musical organization	diversion	facility
communicator	point	club	action	building
scholar	structure	enterprise	group action	movie
alumnus	geographic point	secondary school	change	station
creator	address	business	change of state	conveyance
football player	residence	military unit	game	vehicle
worker	home	team	beginning	device
writer	populated place	senior high school	introduction	work
skilled worker	village	administrative unit	game	publication
artist	settlement	army unit	computer game	terminal
adult	facility	party	vote	railway station
politician	urban area	carrier	election	way
actor	municipality	line	military action	craft
musician	building	league	speech act	broadcasting station

Table 6.1.: Top 20 Subtypes of the 5 Top-Level Types

We are not aware of any similarly rich type hierarchies used in prior work on NER and entity typing. Our approach can easily plug in alternative type taxonomies, either derived from other knowledge bases like Freebase or DBpedia as in [34], or from hand-crafted resources such as WordNet.

6.2.2. Feature Set

For a general approach and for applicability to arbitrary texts, we use only features that are automatically extracted from input texts. We do not use any features that require manual annotations, such as sense-tagging of general words and phrases in training documents. This discriminates our method from some of the prior work which used WordNet senses as features (e.g., [52]). In the following, we briefly discuss each group of features and how they are derived. Table 6.2 summarizes our feature set.

Mention String: We derive four features from the entity mention string. The mention string itself (a noun phrase consisting of one or more consecutive words) is one feature.

The other three features are unigrams, bigrams, and trigrams that overlap with the mention string.

Sentence Surrounding Mention: We also leverage the sentence where the mention appears. Four features are derived from a bounded-size window around the mention: all unigrams, bigrams, and trigrams in the sentence along with their distance to the mention, and all unigrams along with their absolute distance to the mention. Distance captures whether features occur left or right of the mention, whereas absolute distance reflects only the number of words between feature and mention. This set of features is limited by the size of the window around the mention, a parameter of our method. In experiments, we set this to a conservative value of 3 words.

Mention Paragraph: We further consider the entire paragraph within which the mention appears. This may give additional topical cues about the mention type (e.g., if the paragraph talks about a music concert, this is a cue for mapping people names to musician types). We create three features here: unigrams, bigrams, and trigrams without including any distance information. Again, we can control the width of influence of these features, by limiting the size of the window around the mention (truncated at the paragraph boundaries). In our experiments we set the boundaries of the paragraph features to a window of 2000 characters before and 2000 characters after the mention.

Grammatical Features: There are four features of this kind. First, we use part-of-speech tags of the tokens surrounding the mention within a bounded-size window of tokens before and after the mention. The also come in distance, and absolute distance versions. Second and third, we create a feature for the first occurrence of a "he" or "she" pronoun in the same sentence and in the subsequent sentence following the mention, along with the distance to the mention. Finally, we use the closest verb-preposition pair preceding the mention.

Gazetteer Features: For each type of our taxonomy, we build a type-specific gazetteer of words occurring in names of entities of a type. This is derived from the YAGO knowledge base, which has an extensive dictionary of name-entity pairs extracted from Wikipedia redirects and link-anchor texts. An alternative source for this kind of information is the recently released Google corpus of Web links to Wikipedia [47]. We then construct, for each type, a binary feature that indicates if the mention contains a word occurring in this type's gazetteer. Note that this is a fully automated feature construction, and it does by no means determine the mention type(s) already, as most words occur in the gazetteers

Input	Derived Features
Mention String	MENTION UNIGRAM_MEN BIGRAM_MEN TRIGRAM_MEN
Mention Sentence	UNIGRAM_REL UNIGRAM_ABS BIGRAM_REL TRIGRAM_REL
Mention Paragraph	PARA_UNIGRAM PARA_BIGRAM PARA_TRIGRAM
Grammatical Features	POS Stanford NER Tagger FIRST_HE_SHE_SAME_SENT_AFT_MEN_REL FIRST_PRP_HE_SHE_NEXT_SENT_REL LAST_VERB_PREP_TUPLE_BEF_MEN
Gazetteer Features	OCCURS_TYPE1_WORDS OCCURS_TYPE2_WORDS ...

Table 6.2.: Summary of Features Used for Classification

of many different types. For example, "Alice" occurs in virtually every subclass of Person but also in city names like "Alice Springs" and other locations, as well as in songs, movies, and other products or organizations.

6.3. Classifier

6.3.1. Hierarchical Classifier

Based on the feature set defined in the previous section, we build a set of type-specific classifiers using the SVM software liblinear [15, 8]. As our YAGO-based type system integrates WordNet and Wikipedia categories, we obtain ample training data from

Wikipedia effortlessly, simply by following all Wikipedia anchor texts to the corresponding YAGO entities.

For each type, we consider Wikipedia mentions (and their surrounding sentences and paragraphs, see feature model) of the type's instances as positive training samples. For discriminative learning, we use all siblings in the type hierarchy as negative samples. That is, the classifier considers one type against all other types that have the same parent type (e.g., Artist vs. Politician, Athlete, etc. – all under the same parent). As the subclasses of a given type t do not necessarily cover all entities of t, we add a subclass `Others` to each non-leaf type. The positive samples for `Others` are those instances of a type t that do not explicitly belong to any of its subclasses. Conversely, the classifiers for non-leaf nodes include all instances of their subtypes as positive samples (with full weight).

When presented with a new test input, HYENA runs this through the hierarchy of type-specific classifiers in a top-down manner. At each level of the hierarchy, we invoke all classifiers that have the same parent. The test mention is assigned to all types for which the corresponding classifier signals confidence of acceptance or rejection. Only when a mention is accepted for type t, it is presented to all children classifiers of t.

6.3.2. Meta Classifier

The HYENA method outlined above uses a global threshold θ for accepting test mentions to a class. Using a single parameter for all types is not fully satisfying, as different types may exhibit very different characteristics. So the optimal acceptance threshold may be highly type-dependent.

To overcome this limitation, we devised a meta classifier that ranks the types for each test mention by decreasing confidence values and then predicts the "right" number of top-n labels to be assigned to a mention. Our approach to this end follows the methodology of [50]. We use the confidence values of the type-specific classifier ensemble as meta-features, and train a multi-class logistic regression classifier to whose output is a suitable value of n for given features.

We combine the base classifiers and the meta classifier by first running the entire ensemble top-down along the type hierarchy, and then letting the meta model decide on how many of the highest-scoring types we accept for a mention.

6.4. Experiments

In the following subsections we describe the experiments conducted to evaluate HYENA. First, we will introduce the experimental setup. Then we will describe our experiments on different ground truth datasets geared for high precision and high recall. Afterwards we

will introduce results based on employing a meta-classifier in order to improve precision for quality-sensitive use cases. Finally, we analyze the impact of various features employed by HYENA.

6.4.1. Setup

System: The described methods have been implemented in the HYENA framework. The Stanford NLP tools have been used to identify mentions of named entities and to extract the grammatical features from the surrounding context. We used the YAGO2 knowledge base to construct the gazetteer features.
Data: We used the English Wikipedia edition as of 2012-05-02. In order to obtain ground-truth type labels, we exploited the links pointing to other Wikipedia articles by resolving them to their corresponding YAGO2 entity and retrieving their semantic types from YAGO2. For example from the Wikipedia markup

> "In June 1989, Obama met [[Michelle Obama|Michelle Robinson]] when he was employed as a summer associate at the Chicago law firm of [[Sidley Austin]]"

the following YAGO2 entities are determined:

- Michelle Robinson → http://yago-knowledge.org/resource/Michelle_Obama
- Sidley Austin → http://yago-knowledge.org/resource/Sidley_Austin

HYENA is trained on a randomly selected set of 50,000 Wikipedia articles, containing around 1.6 million entity mentions. 92% of the corresponding entities belong to at least one of our 5 top-level types, with 11% belonging to at least two top-level types. Testing of HYENA was performed on 10,000 randomly selected Wikipedia articles, withheld from the same Wikipedia edition and disjoint from the training data. Some properties of training and test data are summarized in Table 6.3. All our experimental data is available on our Web site http://www.mpi-inf.mpg.de/yago-naga/hyena/.
Performance Measures: In order to measure the performance of entity type classification by HYENA, we perform micro- and macro-evaluation of our approach for precision, recall and F1 scores. To this end, we define the measures as follows:
Let T be the set of all types in our hierarchy, and let I_t be the set of instances tagged with the type t, and $\hat{I}_t$ the set of instances that are predicted to be of t. Then, micro-evaluation measures are:

data property	training	testing
# of articles	50,000	10,000
# of instances (all types)	1,613,340	253,029
# of location instances	489,003 (30%)	86,936 (34.4%)
# of person instances	426,467 (26.4%)	62,446 (24.6%)
# of organization instances	219,716 (13.6%)	38,293 (15.1%)
# of artifact instances	204,802 (12.7%)	31,899 (12.6%)
# of event instances	176,549 (10.9%)	28,952 (11.4%)
# instances in 1 top-level class	1,131,994 (70.2%)	179,240 (70.8%)
# instances in 2 top-level classes	182,508 (11.3%)	33,399 (13.2%)
# instances in more than 2 top-level classes	6,492 (0.4%)	828 (0.3%)
# instances not in any class	292,346 (18.1%)	39,562 (15.6%)

Table 6.3.: Properties of Training and Testing Data

$$Precision_{micro} = \frac{\sum_{t \in T} \left| I_t \cap \hat{I}_t \right|}{\sum_{t \in T} \left| \hat{I}_t \right|} \quad and \quad Recall_{micro} = \frac{\sum_{t \in T} \left| I_t \cap \hat{I}_t \right|}{\sum_{t \in T} |I_t|}$$

and macro-evalution measures are:

$$Precision_{macro} = \frac{1}{|T|} \sum_{t \in T} \frac{\left| I_t \cap \hat{I}_t \right|}{\left| \hat{I}_t \right|} \quad and \quad Recall_{macro} = \frac{1}{|T|} \sum_{t \in T} \frac{\left| I_t \cap \hat{I}_t \right|}{|I_t|}$$

Competitors: From literature, we identified those prior methods that target fine-grained, multi-level type classification and used publicly available corpora on which we could run HYENA for direct comparison. These are the methods of [19] referred to as *HOVY*), [52] referred to as *NG*), and *FIGER* by [34]. We preferred experiments on the competitors'

	Macro			Micro		
	Prec.	Rec.	F1	Prec.	Rec.	F1
5 Top-level Types	0.941	0.922	0.932	0.949	0.936	0.943
All 505 Types	0.878	0.863	0.87	0.913	0.932	0.922

Table 6.4.: Overall Experimental Results for HYENA on Wikipedia 10000 articles

datasets to avoid re-implementation and to give our opponents the benefit of their original optimization and tuning.

6.4.2. Multi-label Classification

In the following subsections we will present our multi-label experiments that are geared for high precision and high recall in detail. To this end, we present results against ground truth coming from Wikipedia, the BBN Pronoun Coreference Corpus and Entity Type Corpus (LDC2005T33)[1] and the FIGER-Gold dataset. Baseline for all HYENA experiments are the predictors trained on the 50,000 Wikipedia articles as mentioned above. When applying HYENA to a different dataset than Wikipedia, we will present results for HYENA configurations adopted for those settings as well.

HYENA experiments on Wikipedia

HYENA is trained on a randomly selected set of 50,000 Wikipedia articles, containing around 1.6 million entity mentions. Testing of HYENA was performed on 10,000 randomly selected Wikipedia articles, withheld from the same Wikipedia edition and disjoint from the training data. The results for HYENA are shown in Table 6.4. As can be seen from the table, HYENA achieves very high F1 scores of around 94% for its 5 top-level types. Evaluated against the entire hierarchy, F1 scores are still remarkably high with F1 scores of 87% and 92% for macro and micro evaluations, respectively. The slightly weaker results for the macro evaluation are explainable by our fine-grained hierarchy, which also contains "long-tail types". However, the overall micro results show that these types contain relatively instances only.

[1] http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2005T33

		Macro			Micro		
		Prec.	Rec.	F1	Prec.	Rec.	F1
5 Top-level Types	*HOVY*	0.522	0.464	0.491	0.568	0.51	0.537
	HYENA	**0.941**	**0.922**	**0.932**	**0.949**	**0.936**	**0.943**
All 505 Types	*HOVY*	0.253	0.18	0.21	0.405	0.355	0.378
	HYENA	**0.878**	**0.863**	**0.87**	**0.913**	**0.932**	**0.922**

Table 6.5.: Results of HYENA vs *HOVY* (trained and tested on Wikipedia 10000 articles)

In order to compare against *HOVY*, we emulated their method within the HYENA framework. This was done by specifically configuring the feature set, and using the same training and testing instances as for HYENA. The results of this experiment are shown in Table 6.5. HYENA significantly outperforms *HOVY*. *HOVY* shows decent performance for the 5 top-level types, similar to the original results reported in in [19]. However, it becomes obvious that *HOVY* has not been designed for such a fine-grained type hierarchy as its performance sharply drops for more subtypes at deeper levels.

HYENA Experiments on FIGER-GOLD

The FIGER-GOLD dataset consists of 18 news reports from a university website, as well as local newspapers and specialized magazines. All data was manually annotated based on the type system of 112 types as reported in [34]. The test dataset was annotated with at least one label per mention. This resulted in a total of 434 sentences with 562 entities having 771 labels coming from 42 out of the 112 types.

For comparison against *FIGER* we performed a mapping onto the hierarchy of HYENA. In order to achieve comparability between the levels of types being mapped, we focused on the 5 top-level types. Since the FIGER-GOLD dataset does not come with null mappings, we ran HYENA in two configurations. In the first configuration, HYENA was used without any special consideration as in the previous experiment. Here the classifier was trained to deal with abstract concepts (e.g. `Chinese Philosophy`) that are of generic type `ENTITY_OTHER` only and do not belong to any specific type in our hierarchy. In a second configuration, referred to as "no null tags", we executed HYENA based on the

	Macro			Micro		
	Prec.	Rec.	F1	Prec.	Rec.	F1
FIGER	**0.75**	0.743	0.743	**0.828**	**0.838**	**0.833**
HYENA	0.745	0.631	0.686	0.815	0.645	0.72
HYENA (no null tags)	0.724	**0.801**	**0.75**	0.788	0.814	0.801

Table 6.6.: Results of HYENA vs *FIGER* (trained on Wikipedia and tested on FIGER-Gold)

same classifier as before, but enforced the assignment of at least one label for a specific type.

Results for both configurations are shown in Table 6.6. In the standard configuration, HYENA shows precision scores close to *FIGER*. However, HYENA suffers from the training against abstract concepts. In the second configuration, both systems achieve results in the same range with slight advantages for *FIGER* on micro-average and overall better results of HYENA on macro-average. The overall 10% drop of HYENA's performance compared with the experiments on Wikipedia are due to the nature of the FIGER-GOLD dataset, which comes with short sentences so that context features of HYENA are not that effective. Furthermore, 771 type labels for 562 entity mentions (not entities) is only a very moderate amount of multi-label classification. This is disadvantageous for HYENA, which has been designed for data where number of labels per mention is higher (e.g. Wikipedia).

HYENA Experiments on BBN

The BBN Pronoun Coreference and Entity Type Corpus (LDC2005T33) corpus consists of 2311 manually annotated documents, including entity type classification. Since *NG* exploits the word sense number for disambiguation, the corpus is restricted to those 200 documents (160 training, 40 testing) that have corresponding annotations.

For comparison against *NG* we performed a mapping onto the hierarchy of HYENA. [52] reports results for 16 types for the *NG* data. These types include 8 non-entity types: `Date`, `Money`, `Percent`, `Quantity`, `Cardinal`, `NORP` [Nationality, Religion, etc.], `Ordinal` and `Substance`. Furthermore, there are 5 descriptor types (`_DESC`) that cannot

	Macro			Micro		
	Prec.	Rec.	F1	Prec.	Rec.	F1
NG (trained on BBN)	0.859	0.864	0.862	0.812	0.871	0.84
HYENA (trained on Wikipedia)	**0.943**	0.406	0.568	**0.932**	0.371	0.531
HYENA (trained on Wikipedia, no null tags)	0.818	0.671	0.737	0.835	0.632	0.719
HYENA (trained on BBN)	0.916	**0.909**	**0.911**	0.919	**0.881**	**0.899**

Table 6.7.: Results of HYENA vs *NG* (tested on BBN Corpus)

be mapped. This resulted in mapping the 3 top-level types: `Person`, `Organization` and `GPE` (country, city, states, etc.). As before, the BBN corpus does not come with null mappings and the *NG* method has been trained on a different corpus, we ran HYENA in three configurations. In the first configuration ("trained on Wikipedia"), HYENA was used as it is. In the second configuration ("trained on Wikipedia, no null mapping"), we enforced at least one type label to be assigned as in the previous experiment. Finally, we also trained HYENA on the *NG* training set in the third configuration ("trained on BBN").

Results for experiments on the BBN dataset are shown in Table 6.7. HYENA exhibits high precision already with its standard configuration. However, it suffers from low recall in this setting, which is due to its training against abstract concepts that result in not assigning any type. When enforcing HYENA to assign at least one tag, we recognize a strong improvement in F1 scores similar to the experiments on FIGER-Gold. In the third configuration, the fairest side-by-side comparison, we clearly outperform *NG*.

6.4.3. Meta-Classification

In application use-cases for type labeling, such as named entity disambiguation, precision is often more important than recall. This is particularly critical and demanding for types that suffer from data sparsity (less prominent and/or less populated types) on the deeper levels of the type hierarchy. For example, when disambiguating named entities, it may be crucial to distinguish a `Painter` from a `Musician`. In order to improve the precision of fine-grained type labeling, we applied a meta-classifier as described in Section 6.3.2. The meta-classifier adjusts, per level, the threshold for the number of types that an

Technique		Macro			Micro		
		Prec.	Rec.	F1	Prec.	Rec.	F1
All 505	HYENA	0.878	**0.863**	**0.87**	0.913	**0.932**	**0.922**
Types	HYENA + meta-classifier	**0.89**	0.837	0.862	**0.916**	0.914	0.915

Table 6.8.: Performance gain in precision by meta-classifaction

	Macro			Micro		
	Prec.	Rec.	F1	Prec.	Rec.	F1
HYENA	0.673	**0.638**	**0.644**	0.659	**0.681**	**0.67**
HYENA + meta-classifier	**0.693**	0.619	0.638	**0.674**	0.66	0.667

Table 6.9.: Meta-classifier impact on the 5% worst-performing classes

individual mention should have. Typically this results in a more conservative behavior of the classifier.

Table 6.8 shows the overall performance gain achieved by applying HYENA's meta-classifier. As can be seen from the table, meta classification helps to improve the macro-precision over all 505 types by more than 1%. When specifically focusing on the 5% types that performed worst with the HYENA base classifiers, we even gained more than 2% in precision, as shown in Table 6.9. The top-5 winners in this group are (in ascending order) `Tour`, `Pageant`, `Presentation`, `Battalion` and `Ghost Town` with performance gains ranging from 5% up to 13%.

6.4.4. HYENA Feature Analysis

In addition to a comprehensive set of features, the HYENA method exploits two additional assets that contribute to its very good performance: a large amount of training data derived from the linkage between Wikipedia and YAGO, and the gazetteer features

Size of training set (# of articles)	5 Top-level Types			All 505 Types		
	Prec.	Rec.	F1	Prec.	Rec.	F1
50,000	0.949	0.936	0.942	0.913	0.932	0.922
20,000	0.937	0.924	0.93	0.893	0.917	0.905
10,000	0.929	0.915	0.922	0.881	0.907	0.894
5,000	0.92	0.903	0.912	0.869	0.89	0.879
50,000 (without gazetteers)	0.915	0.825	0.868	0.82	0.718	0.766

Table 6.10.: Micro-average impact of varying the number of Wikipedia articles used for training

derived from YAGO. To assess the impact of each of these two extra assets, we ran experiments with varying numbers of training instances and with enabling or disabling gazetteer features. The influence of the number of training instance is shown in Table 6.10. Precision and recall improve with adding more training data, and the improvement is most significant for the deeper levels of the type hierarchy where sparseness is a concern. When gazetteer features are disabled, the performance dropped significantly, which shows the strong benefit from the gazetteers.

6.5. Extrinsic Study on Named Entity Disambiguation

In this section we present an extrinsic study on harnessing HYENA for named entity disambiguation (NED). Specifically, we consider a state-of-the-art NED tool, AIDA, provided by the authors of [26]. This NED method uses a combination of contextual similarity and entity-entity coherence for joint inference on how to map a set of entity mentions in an input text onto canonical entities registered in a knowledge base. It uses advanced graph algorithms which are computationally expensive. Alternative methods with similarly strong results would be based on machine learning with probabilistic factor graph which is equally if not more expensive. Therefore, it is desirable to prune the search space of potentially relevant candidate entities as much as possible and as early as possible.

Threshold	% dropped entities	% of unsolvable mentions	Avg. Document Precision	Avg. Mention Precision
0.0	49.2	16.1	0.659	0.639
−0.5	45.7	12.3	0.738	0.713
−1.0	37.9	7.6	0.781	0.76
−1.5	28.8	4.7	0.791	0.779
−2.0	22.3	3.1	0.8	0.792
−2.5	17.7	2.2	0.802	0.798
AIDA	0	0	0.82	0.823

Table 6.11.: Impact of Varying Type Prediction Confidence Threshold on NED Results

In the following experiment, we use the type predictions by HYENA to identify candidate entities that are unlikely to be among the true entities for the given mentions. For example, for a sentence like "He was born in Victoria", once we restrict the possible types of the mention "Victoria" to `Location` or perhaps even to `City` and `Region`, we could drop all entities of type `Person` or `River` and `Lake` from the candidate space and thus speed up the NED computation.

We use the confidence scores of HYENA to remove entities of types with type scores below some threshold θ. Our technique proceeds in three steps:

1. Invoke HYENA on the mention to obtain the predicted types for this mention as well as their confidence scores.
2. Generate entity candidates using AIDA and its underlying name-entity dictionary.
3. For each candidate, if there is no overlap between the entity types and the predicted mention types with confidence greater than or equal to θ, drop the candidate.
4. Run AIDA on the reduced candidate space.

When dropping entities from the candidate space, there is a risk of dropping the correct entity. In this case, we consider the mention as *unsolvable*. We study the effect of varying the relaxation parameter θ on the fraction of dropped entities (i.e., the search space

reduction) and the fraction of mentions that are rendered *unsolvable*. We varied θ from -2.5 up to 0 with step size 0.5, and also compared to the variant without any pruning ($\theta = -\infty$) We performed our experiment on the extended CoNLL 2003 NER dataset with manual entity annotations from [26]. The results are shown in Table 6.11.

We see that with a pruning threshold of $\theta = -1$, we can prune almost 40% while rendering less than 8% of the mentions unsolvable. Also, the overall precision of the NED results drops only by a small amount compared to the variant without pruning. This holds for both averaging over all documents and averaging over all mentions. The search space reduction of 40% actually results in a much larger saving in run-time because the graph algorithm that AIDA uses for NED has super-linear complexity (NP-hard in the worst case, but typically $O(n \log n)$ or $O(n^2)$ with appropriate approximation algorithms.

6.6. System Implementation

6.6.1. Overview

As described in Section 6.2, HYENA classifies mentions of named entities onto a hierarchy of 505 types using large set of features. A random subset of the English Wikipedia has been used for training HYENA. By exploiting Wikipedia anchor links, mentions of named entities are automatically disambiguated to their correct entities. Each Wikipedia named entity has a corresponding YAGO entity labeled with an accurate set of types, and hence we effortlessly obtain a huge training data set (cf. data properties in Table 6.3).

We build type-specific classifiers using the SVM software LIBLINEAR (cf. `http://liblinear.bwaldvogel.de/`). Each model comes with a comprehensive feature set. While larger models (with more features) improve the accuracy, they significantly affect the applicability of the system. A single model file occupies around 150MB disk space leading to a total of 84.7GB for all models. As a consequence, there is a substantial setup time to load all models in memory and a high-memory server (48 cores with 512GB of RAM) is required for computation. An analysis showed that each single feature contributes to the overall performance of HYENA, but only a tiny subset of all features is relevant for a single classifier. Therefore, most of the models are extremely sparse.

6.6.2. Sparse Models Representation

There are several workarounds applicable to batch mode operations, e.g. by performing classifications per level only. However, this is not an option for on-the-fly computations. For that reason we opted for a sparse-model representation.

LIBLINEAR model files are normalized textual files: a header (data about the model and the total number of features), followed by listing the weights assigned to each feature (line number indicates the feature ID). Each model file has been post-processed to produce 2 files:

- A compacted model file containing only features of non-zero weights. Its header reflects the reduced number of features.
- A meta-data file. It maps the new features IDs to the original feature IDs.

Due to the observed sparsity in the model files, particularly at deeper levels, there is a significant decrease in disk space consumption for the compacted model files and hence in the memory requirements.

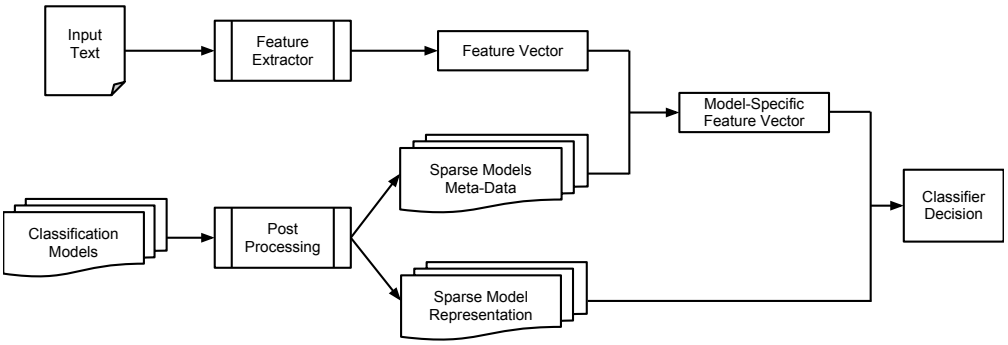

Figure 6.2.: Modified system architecture designed for handling sparse models

6.6.3. Sparse Models Classification

By switching to the sparse model representation the architecture of the whole system is affected. In particular, modified versions of feature vectors need to be generated for each classifier; this is because a lot of features have been omitted from specific classifiers (those with zero weights). Consequently, the feature IDs need to be mapped to the new feature space of each classifier. The conceptual design of the new architecture is illustrated in Figure 6.2.

7. Conclusion

7.1. Contributions

This dissertation addressed the problem of building a customizable Named Entity Recognition and Disambiguation (NERD) framework. The contributions are summarized as follows.

The first contribution is U-AIDA: a universal architecture for building NERD solutions. U-AIDA supports building a multi-knowledge base NERD solution. In addition, it provides downstream applications with run-time customization by enabling and disabling various data components.

The second contribution is the development of two domain-specific NERD solutions within the U-AIDA framework. The first is a system for disambiguating German documents in German National Library (DNB). The underlying entity catalog has been obtained from the general knowledge resource, German Wikipedia, together with a specialized KB created by the DNB. The second system is a NERD solution for disambiguating tweets. While the first system shows the power of U-AIDA with respect to integrating different knowledge resources, the second highlights the U-AIDA customizability to address a different style of text.

The third contribution is a complete pipeline for building non-English NERD solutions. We focused on languages with poor entity-annotated resources such as Arabic. We leverage external dictionaries as well as statistical machine translation to enrich the NERD repository with cross-language evidences. Experiments showed that our enrichment techniques delivered improvements in both precision and recall.

The fourth contribution is HYENA: a novel method for fine-grained type classification of entity mentions. HYENA is essential to classify mentions of entities that do not exist in the underlying KB. In contrast to prior methods, we can deal well with hundreds of types in a multi-level hierarchy, and consider that a mention can have many different types – a situation that does not (likely) occur in prior work with 10 to 100 types on merely two different levels. In the presented experiments, HYENA outperformed the baseline competitors even on their original datasets.

7.2. Outlook

While this dissertation addressed a number of key problems, more research should be conducted to address the following research issues.

7.2.1. Adaptive U-AIDA

This work provided a customizable and flexible architecture that can be configured according to the application requirements. However, it is designed assuming all configuration will be manually preformed by the application architects. It would be useful to build a smart system that can automatically adapt itself to the input text and switch on and off different components accordingly. For example, currently one disambiguation technique is applied on the whole input text. However, the system should automatically detect when coherence should be applied to subsets of the mentions of the input text. In addition, as discussed in this work, different entries in the name-entity dictionary and the entity keyphrases catalog exhibit different qualities. The system should be able to first use the data of highest quality and then consider less accurate data if needed.

7.2.2. Multi-Genre Joint NERD

We addressed the problem of disambiguating text with entities originating from different domains by considering multi-KBs when building the NERD solution. However, we did not consider the case when the inputs belong to different genres. For example, collectively disambiguating a professional homepage of a person together with his most recent tweets and Facebook profile, or disambiguating a book together with the biographies of the authors. The intuitive idea here is that cross-domain evidence could improve the quality.

7.2.3. Disambiguating Comparable Corpora

This work addressed the problem of a NERD system for different languages. We used cross-language evidence to enrich the underlying KB. However, the input text is assumed to be of one language. Using cross-language evidence might also help in improving the disambiguation quality. For example, performing a joint disambiguation of news articles of different languages covering the same event, or disambiguating a book and its translation in another language.

7.2.4. Hybrid Named Entity Classification

In this dissertation, we developed a machine learning based classification system. While we used a comprehensive set of features, the same technique has been applied to all classes. Other approaches, such as PEARL [42], use salient patterns to classify entities. One approach cannot handle all types of classes equally well. For example, "graduated from" is a salient pattern to classify the current entity as a student with probably higher precision than our approach. However, no trivial pattern can distinguish between football players and basketball players. It is important to build a hybrid system that leverages different techniques at different nodes in the type hierarchy.

Bibliography

[1] Yaser Al-Onaizan and Kevin Knight. Translating named entities using monolingual and bilingual resources. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, ACL '02, pages 400–408, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[2] Enrique Alfonseca and Suresh Manandhar. An unsupervised method for general named entity recognition and automated concept discovery. In *Proceedings of the 1st International Conference on General WordNet*, India, 2002.

[3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *Processdings of the 6th International Semantic Web Conference*, ISWC '07, pages 11–15. Springer, 2007.

[4] Mahmoud Azab, Houda Bouamor, Behrang Mohit, and Kemal Oflazer. Dudley north visits north london: Learning when to transliterate to arabic. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '13, pages 439–444, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[5] Artem Boldyrev. Dictionary-based named entity recognition. Master's thesis, Universität des Saarlandes, Saarbrücken, 2013.

[6] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA, 2008. ACM.

[7] Amparo Elizabeth Cano, Giuseppe Rizzo, Andrea Varga, Matthew Rowe, Milan Stankovic, and Aba-Sah Dadzie. #microposts2014 neel challenge: Measuring the performance of entity linking systems in social streams. In *Proceedings of the #Microposts2014 NEEL Challenge*, 2014.

[8] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM TIST*, 2(3):27, 2011.

[9] Silviu Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[10] Hamish Cunningham. Gate, a general architecture for text engineering. *Computers and the Humanities*, 36(2):223–254, 2002.

[11] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, ACL'02, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[12] Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[13] Asif Ekbal, Eva Sourjikova, Anette Frank, and Simone P. Ponzetto. Assessing the challenge of fine-grained named entity recognition and classification. In *Proceedings of the Named Entities Workshop*, NEWS '10, pages 93–101, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[14] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[15] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[16] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

[17] Paolo Ferragina and Ugo Scaiella. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM International*

Conference on Information and Knowledge Management, CIKM '10, pages 1625–1628, New York, NY, USA, 2010. ACM.

[18] 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.

[19] Michael Fleischman and Eduard Hovy. Fine grained classification of named entities. In *Proceedings of the 19th international conference on Computational linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[20] Mohamed H. Gadelrab, Mohamed Amir Yosef, and Gerhard Weikum. Named entity disambiguation for resource-poor languages. In *Proceedings of the 8th International Workshop on Exploiting Semantic Annotations in Information Retrieval*, ESAIR '15. ACM, 2015.

[21] Claudio Giuliano. Fine-grained classification of named entities exploiting latent semantic kernels. In *Proceedings of the 13th Conference on Computational Natural Language Learning*, CoNLL '09, pages 201–209, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[22] Ondrej Hálek, Rudolf Rosa, Ales Tamchyna, and Ondrej Bojar. Named entities from wikipedia for machine translation. In *Prceeding the Conference on Information Technologies - Applications and Theory*, ITAT '11, pages 23–30, Košice, Slovakia, 2011. Univerzita Pavla Jozefa Šafárika.

[23] Johannes Hoffart, Yasemin Altun, and Gerhard Weikum. Discovering Emerging Entities with Ambiguous Names. In *Proceedings of the 23rd International World Wide Web Conference*, WWW '14, pages 385–395, 2014.

[24] Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. KORE: Keyphrase Overlap Relatedness for Entity Disambiguation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 545–554, 2012.

[25] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61, January 2013.

[26] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 782–792, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[27] Richard Holt. Twitter in numbers, March 2013.

[28] Fei Huang, Stephan Vogel, and Alex Waibel. Improving named entity translation combining phonetic and semantic similarities. In *Processdings of the Conference on Human Language Technology / North American chapter of the Association for Computational Linguistics*, volume 2004 of *NAACL-HLT '04'*, pages 281–288, 2004.

[29] Yusra Ibrahim, Mohamed Amir Yosef, and Gerhard Weikum. Aida-social: Entity linking on the social stream. In *Proceedings of the 7th International Workshop on Exploiting Semantic Annotations in Information Retrieval*, ESAIR '14, pages 17–19, New York, NY, USA, 2014. ACM.

[30] Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 457–466, New York, NY, USA, 2009. ACM.

[31] Young-Suk Lee. Confusion network for arabic name disambiguation and transliteration in statistical machine translation. In *Proceedings of the 25th International Conference on Computational Linguistics*, COLING '14, pages 433–443, 2014.

[32] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, SIGDOC '86, pages 24–26, New York, NY, USA, 1986. ACM.

[33] et al. Li, Xuansong. Gale arabic-english word alignment training part 1– newswire and web ldc2014t05, 2014.

[34] Xiao Ling and Daniel S. Weld. Fine-grained entity recognition. In *Processings of AAAI Conference on Artificial Intelligence*, AAAI '12, 2012.

[35] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *Proceedings of the Conference on Innovative Data Systems Research*, CIDR '15. www.cidrdb.org, 2015.

[36] James Mayfield, Dawn Lawrie, Paul McNamee, and Douglas W. Oard. Building a cross-language entity linking collection in twenty-one languages. In *Proceedings of the 2nd Conference of the Cross-Language Evaluation Forum*, CLEF '11', pages 3–13. Springer, 2011.

[37] Paul McNamee, James Mayfield, Dawn Lawrie, Douglas W Oard, and David S Doermann. Cross-language entity linking. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, IJCNLP '11, pages 255–263, 2011.

[38] Pablo N. Mendes, Max Jakob, Andres Garcia-Silva, and Christian Bizer. Dbpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, 2011.

[39] David Milne and Ian H. Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 509–518, New York, NY, USA, 2008. ACM.

[40] Andrea Moro, Alessandro Raganato, and Roberto Navigli. Entity Linking meets Word Sense Disambiguation: a Unified Approach. *Transactions of the Association for Computational Linguistics*, 2:231–244, 2014.

[41] Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. Scalable knowledge harvesting with high precision and high recall. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 227–236, New York, NY, USA, 2011. ACM.

[42] Ndapandula Nakashole, Tomasz Tylenda, and Gerhard Weikum. Fine-grained semantic typing of emerging entities. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL '13, pages 1488–1497, 2013.

[43] 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.

[44] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the 13th Conference on Computational Natural Language Learning*, CoNLL '09, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[45] Lev-Arie Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual*

Meeting of the Association for Computational Linguistics, ACL '11, pages 1375–1384, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[46] Nakatani Shuyo. Language detection library for java, 2010.

[47] Valentin I. Spitkovsky and Angel X. Chang. A cross-lingual dictionary for english wikipedia concepts. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, LREC '12, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA).

[48] Ralf Steinberger, Bruno Pouliquen, Mijail Alexandrov Kabadjov, and Erik Van der Goot. Jrc-names: A freely available, highly multilingual named entity resource. *CoRR*, abs/1309.6162, 2013.

[49] Fabian Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A core of semantic knowledge - unifying WordNet and Wikipedia. In *Proceedings of the 16th International World Wide Web Conference*, WWW '07, pages 697–706, Banff, Canada, 2007. ACM.

[50] Lei Tang, Suju Rajan, and Vijay K. Narayanan. Large scale multi-label classification via metalabeler. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 211–220, New York, NY, USA, 2009. ACM.

[51] Grigorios Tsoumakas, Min-Ling Zhang, and Zhi-Hua Zhou. Introduction to the special issue on learning from multi-label data. *Machine Learning*, 88(1-2):1–4, 2012.

[52] Md. Altaf ur Rahman and Vincent Ng. Inducing fine-grained semantic classes via hierarchical and collective classification. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 931–939, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[53] Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Michael Röder, Daniel Gerber, SandroAthaide Coelho, Sören Auer, and Andreas Both. AGDISTIS - Graph-Based Disambiguation of Named Entities Using Linked Data. In *Proceedings of the Interntaional Conference on The Semantic Web*, volume 8796 of *ISWC '14*, pages 457–471. Springer International Publishing, 2014.

[54] Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Röder Michael, Sören Auer, Daniel Gerber, and Andreas Both. Agdistis - agnostic disambiguation of named entities using linked open data. In *European Conference on Artificial Intelligence*, page 2. IOS Press, 2014.

[55] Nina Wacholder, Yael Ravin, and Misook Choi. Disambiguation of proper names in text. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, ANLC '97, pages 202–208, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics.

[56] Gerhard Weikum, Johannes Hoffart, Ndapandula Nakashole, Marc Spaniol, Fabian Suchanek, and Mohamed Amir Yosef. Big data methods for computational linguistics. *IEEE Data Engineering Bulletin*, 35(3):46–55, 2012.

[57] Jonathan Wright, Kira Griffitt, Joe Ellis, Stephanie Strassel, and Brendan Callahan. Annotation trees: Ldc's customizable, extensible, scalable, annotation infrastructure. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, LREC '12, pages 479–485, 2012.

[58] Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart Marc Spaniol, and Gerhard Weikum. HYENA: Hierarchical Type Classification for Entity Names. In *Proceedings of the 24th International Conference on Computational Linguistics*, COLING '12, pages pp. 1361–1370, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[59] Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart Marc Spaniol, and Gerhard Weikum. HYENA-live: Fine-Grained Online Entity Type Classification from Natural-language Text. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL '13, pages 133–138, Stroudsburg, PA, USA, 2013. Association for Computational Linguistics.

[60] Mohamed Amir Yosef, Johannes Hoffart, Ilaria Bordino, Marc Spaniol, and Gerhard Weikum. AIDA: an online tool for accurate disambiguation of named entities in text and tables. *Proceedings of the Very Large Scale Data Bases Endowment*, 4(12):1450–1453, 2011.

[61] Mohamed Amir Yosef, Johannes Hoffart, Yusra Ibrahim, Artem Boldyrev, and Gerhard Weikum. Adapting AIDA for tweets. In *Proceedings of the 4th Workshop on Making Sense of Microposts co-located with the 23rd International World Wide Web Conference*, pages 68–69, 2014.

[62] Mohamed Amir Yosef, Marc Spaniol, and Gerhard Weikum. AIDArabic: A named-entity disambiguation framework for Arabic text. In *Proceedgins of the 1st Workshop on Arabic Natural Language Processing*, ANLP '14), pages 187–195, Dohar, Qatar, 2014. ACL.

A. HYENA Type Hierarchy

```
 -- wordnet_entity_100001740
| -- wordnet_artifact_100021939
|  | -- wordnet_instrumentality_103575240
|  |  | -- wordnet_conveyance_103100490
|  |  |  | -- wordnet_vehicle_104524313
|  |  |  |  | -- wordnet_military_vehicle_103764276
|  |  |  |  |  | -- wordnet_warship_104552696
|  |  |  |  |  |  | -- wordnet_destroyer_103180504
|  |  |  |  |  |  | -- wordnet_submersible_104348184
|  |  |  |  |  |  |  | -- wordnet_submarine_104347754
|  |  |  |  | -- wordnet_craft_103125870
|  |  |  |  |  | -- wordnet_vessel_104530566
|  |  |  |  |  |  | -- wordnet_ship_104194289
|  |  |  |  |  |  |  | -- wordnet_cargo_ship_102965300
|  |  |  |  |  |  |  | -- wordnet_warship_104552696
|  |  |  |  |  |  |  |  | -- wordnet_destroyer_103180504
|  |  |  |  |  |  |  |  | -- wordnet_submersible_104348184
|  |  |  |  |  |  |  |  |  | -- wordnet_submarine_104347754
|  |  |  |  |  |  |  | -- wordnet_shipwreck_104197110
|  |  |  |  |  |  | -- wordnet_boat_102858304
|  |  |  |  |  | -- wordnet_aircraft_102686568
|  |  |  |  | -- wordnet_wheeled_vehicle_104576211
|  |  |  |  |  | -- wordnet_self02dpropelled_vehicle_104170037
|  |  |  |  |  |  | -- wordnet_motor_vehicle_103791235
|  |  |  |  |  |  |  | -- wordnet_car_102958343
|  |  |  |  |  |  | -- wordnet_locomotive_103684823
|  |  | -- wordnet_medium_106254669
|  |  |  | -- wordnet_album_106591815
|  |  |  | -- wordnet_print_media_106263609
|  |  |  |  | -- wordnet_press_106263369
|  |  |  |  |  | -- wordnet_magazine_106595351
|  |  |  |  |  | -- wordnet_newspaper_106267145
|  |  | -- wordnet_device_103183080
|  |  |  | -- wordnet_instrument_103574816
|  |  |  |  | -- wordnet_weapon_104565375
```

```
|  |  |  |  |  | -- wordnet_gun_103467984
|  |  |  |  |  |  | -- wordnet_firearm_103343853
|  |  |  | -- wordnet_memory_device_103744840
|  |  |  |  | -- wordnet_recording_104063868
|  |  |  |  |  | -- wordnet_sound_recording_104262678
|  |  |  |  |  |  | -- wordnet_soundtrack_104262969
|  |  |  | -- wordnet_machine_103699975
|  |  |  |  | -- wordnet_computer_103082979
|  |  |  |  |  | -- wordnet_web_site_106359193
|  |  |  |  | -- wordnet_motor_103789946
|  |  |  |  |  | -- wordnet_engine_103287733
|  |  |  | -- wordnet_mechanism_103738472
|  |  |  |  | -- wordnet_mechanical_device_103736970
|  |  |  | -- wordnet_musical_instrument_103800933
|  |  | -- wordnet_container_103094503
|  |  |  | -- wordnet_wheeled_vehicle_104576211
|  |  |  |  | -- wordnet_selfÕ02dpropelled_vehicle_104170037
|  |  |  |  |  | -- wordnet_motor_vehicle_103791235
|  |  |  |  |  |  | -- wordnet_car_102958343
|  |  |  |  |  | -- wordnet_locomotive_103684823
|  |  | -- wordnet_equipment_103294048
|  |  |  | -- wordnet_electronic_equipment_103278248
|  |  |  |  | -- wordnet_set_104176528
|  |  |  |  |  | -- wordnet_receiver_104060647
|  |  | -- wordnet_system_104377057
|  | -- wordnet_structure_104341686
|  |  | -- wordnet_building_102913152
|  |  |  | -- wordnet_place_of_worship_103953416
|  |  |  |  | -- wordnet_church_103028079
|  |  |  |  | -- wordnet_temple_104407435
|  |  |  | -- wordnet_skyscraper_104233124
|  |  |  | -- wordnet_house_103544360
|  |  |  |  | -- wordnet_residence_104079244
|  |  |  |  |  | -- wordnet_religious_residence_104073948
|  |  |  |  |  |  | -- wordnet_monastery_103781244
|  |  |  |  | -- wordnet_mansion_103719053
|  |  |  |  |  | -- wordnet_palace_103878066
|  |  |  | -- wordnet_medical_building_103739518
|  |  |  |  | -- wordnet_hospital_103540595
|  |  |  | -- wordnet_hotel_103542333
|  |  |  | -- wordnet_theater_104417809
|  |  | -- wordnet_memorial_103743902
```

```
|   |   | -- wordnet_housing_103546340
|   |   |   | -- wordnet_dwelling_103259505
|   |   |   |   | -- wordnet_house_103544360
|   |   |   |   |   | -- wordnet_residence_104079244
|   |   |   |   |   |   | -- wordnet_religious_residence_104073948
|   |   |   |   |   |   |   | -- wordnet_monastery_103781244
|   |   |   |   |   | -- wordnet_mansion_103719053
|   |   |   |   |   |   | -- wordnet_palace_103878066
|   |   | -- wordnet_building_complex_102914991
|   |   |   | -- wordnet_plant_103956922
|   |   | -- wordnet_establishment_103297735
|   |   |   | -- wordnet_place_of_business_103953020
|   |   |   |   | -- wordnet_mercantile_establishment_103748162
|   |   |   |   |   | -- wordnet_plaza_103965456
|   |   |   | -- wordnet_institution_103574555
|   |   |   |   | -- wordnet_penal_institution_103907654
|   |   |   |   |   | -- wordnet_correctional_institution_103111690
|   |   | -- wordnet_stadium_104295881
|   |   | -- wordnet_obstruction_103839993
|   |   |   | -- wordnet_barrier_102796623
|   |   |   |   | -- wordnet_dam_103160309
|   |   | -- wordnet_area_102735688
|   |   |   | -- wordnet_room_104105893
|   |   |   |   | -- wordnet_library_103660909
|   |   | -- wordnet_tower_104460130
|   |   | -- wordnet_bridge_102898711
|   | -- wordnet_facility_103315023
|   |   | -- wordnet_station_104306080
|   |   |   | -- wordnet_terminal_104412901
|   |   |   |   | -- wordnet_railway_station_104049098
|   |   |   | -- wordnet_broadcasting_station_102903405
|   |   |   |   | -- wordnet_radio_station_104044119
|   |   |   |   | -- wordnet_television_station_104406350
|   |   |   |   |   | -- wordnet_channel_103006398
|   |   |   | -- wordnet_power_station_103996655
|   |   | -- wordnet_military_installation_103763133
|   |   |   | -- wordnet_military_post_103763403
|   |   |   |   | -- wordnet_garrison_103420559
|   |   | -- wordnet_depository_103177349
|   |   |   | -- wordnet_museum_103800563
|   |   | -- wordnet_airfield_102687992
|   |   |   | -- wordnet_airport_102692232
```

```
|  | -- wordnet_creation_103129123
|  |  | -- wordnet_product_104007894
|  |  |  | -- wordnet_end_product_103287178
|  |  |  |  | -- wordnet_oeuvre_103841417
|  |  |  | -- wordnet_work_104599396
|  |  |  |  | -- wordnet_publication_106589574
|  |  |  |  |  | -- wordnet_book_106410904
|  |  |  |  |  |  | -- wordnet_reference_book_106417598
|  |  |  |  |  | -- wordnet_periodical_106593296
|  |  |  |  |  | -- wordnet_magazine_106595351
|  |  |  | -- wordnet_movie_106613686
|  |  |  |  | -- wordnet_musical_107019172
|  |  |  |  | -- wordnet_documentary_106616806
|  |  | -- wordnet_art_102743547
|  |  |  | -- wordnet_graphic_art_103453809
|  |  |  |  | -- wordnet_painting_103876519
|  | -- wordnet_article_100022903
|  |  | -- wordnet_ware_104550840
|  |  |  | -- wordnet_tableware_104381994
|  |  |  |  | -- wordnet_crockery_103133538
|  | -- wordnet_commodity_103076708
|  |  | -- wordnet_consumer_goods_103093574
|  |  |  | -- wordnet_clothing_103051540
|  | -- wordnet_way_104564698
|  |  | -- wordnet_road_104096066
|  |  |  | -- wordnet_highway_103519981
|  |  |  |  | -- wordnet_expressway_103306610
|  |  |  | -- wordnet_thoroughfare_104426618
|  |  |  |  | -- wordnet_street_104334599
|  |  | -- wordnet_passage_103895293
|  | -- wordnet_covering_103122748
|  |  | -- wordnet_clothing_103051540
| -- wordnet_event_100029378
|  | -- wordnet_act_100030358
|  |  | -- wordnet_action_100037396
|  |  |  | -- wordnet_change_100191142
|  |  |  |  | -- wordnet_change_of_state_100199130
|  |  |  |  |  | -- wordnet_termination_100209943
|  |  |  |  |  |  | -- wordnet_killing_100219012
|  |  |  |  |  |  |  | -- wordnet_homicide_100220023
|  |  |  |  |  |  |  |  | -- wordnet_murder_100220522
|  |  |  |  |  |  |  |  |  | -- wordnet_slaughter_100223983
```

```
|  |  |  |  |  | -- wordnet_beginning_100235435
|  |  |  |  |  |  | -- wordnet_introduction_100238022
|  |  |  |  | -- wordnet_motion_100279835
|  |  |  |  |  | -- wordnet_travel_100295701
|  |  |  |  |  |  | -- wordnet_journey_100306426
|  |  |  |  |  |  |  | -- wordnet_tour_100310666
|  |  |  |  | -- wordnet_motion_100331950
|  |  | -- wordnet_activity_100407535
|  |  |  | -- wordnet_game_100455599
|  |  |  | -- wordnet_diversion_100426928
|  |  |  |  | -- wordnet_sport_100523513
|  |  |  |  | -- wordnet_celebration_100428000
|  |  |  |  |  | -- wordnet_festival_100517728
|  |  |  |  |  |  | -- wordnet_film_festival_100517418
|  |  |  |  | -- wordnet_entertainment_100429048
|  |  |  |  |  | -- wordnet_show_100520257
|  |  |  |  | -- wordnet_game_100430606
|  |  |  |  |  | -- wordnet_computer_game_100458890
|  |  |  | -- wordnet_representation_100898518
|  |  |  |  | -- wordnet_pageant_100899761
|  |  |  | -- wordnet_use_100947128
|  |  |  |  | -- wordnet_application_100949134
|  |  |  |  |  | -- wordnet_technology_100949619
|  |  |  | -- wordnet_wrongdoing_100732746
|  |  |  |  | -- wordnet_transgression_100745005
|  |  |  |  |  | -- wordnet_crime_100766234
|  |  |  | -- wordnet_procedure_101023820
|  |  |  |  | -- wordnet_rule_105846932
|  |  |  |  |  | -- wordnet_algorithm_105847438
|  |  |  | -- wordnet_work_100575741
|  |  |  |  | -- wordnet_investigation_100633864
|  |  |  |  |  | -- wordnet_examination_100635850
|  |  |  |  |  |  | -- wordnet_survey_100644503
|  |  |  |  | -- wordnet_undertaking_100795720
|  |  |  |  | -- wordnet_service_100577525
|  |  |  | -- wordnet_sensory_activity_100876737
|  |  |  |  | -- wordnet_sensing_100876874
|  |  |  |  |  | -- wordnet_look_100877127
|  |  |  |  |  |  | -- wordnet_observation_100879759
|  |  |  | -- wordnet_operation_100955060
|  |  |  |  | -- wordnet_attack_100972621
|  |  |  | -- wordnet_occupation_100582388
```

```
|  |  |  |  | -- wordnet_profession_100609953
|  |  |  |  |  | -- wordnet_technology_100949619
|  |  | -- wordnet_speech_act_107160883
|  |  |  | -- wordnet_description_107201365
|  |  |  |  | -- wordnet_label_107202579
|  |  |  | -- wordnet_disagreement_107180787
|  |  |  |  | -- wordnet_dispute_107181935
|  |  |  |  |  | -- wordnet_controversy_107183151
|  |  |  | -- wordnet_informing_107212190
|  |  |  |  | -- wordnet_report_107217924
|  |  |  |  |  | -- wordnet_gossip_107223170
|  |  |  | -- wordnet_command_107168131
|  |  |  |  | -- wordnet_order_107168623
|  |  | -- wordnet_group_action_101080366
|  |  |  | -- wordnet_vote_100182213
|  |  |  |  | -- wordnet_election_100181781
|  |  |  | -- wordnet_military_action_100952963
|  |  |  |  | -- wordnet_war_100973077
|  |  |  |  | -- wordnet_battle_100953559
|  |  |  |  |  | -- wordnet_naval_battle_100958477
|  |  |  | -- wordnet_social_control_101123598
|  |  |  | -- wordnet_conflict_100958896
|  |  | -- wordnet_communication_106252138
|  | -- wordnet_social_event_107288639
|  |  | -- wordnet_contest_107456188
|  |  |  | -- wordnet_tournament_107464725
|  |  |  | -- wordnet_championship_107457834
|  |  |  | -- wordnet_match_107470671
|  |  |  | -- wordnet_race_107472657
|  |  |  | -- wordnet_race_107458453
|  |  |  |  | -- wordnet_horse_race_107461411
|  |  |  | -- wordnet_game_100456199
|  |  | -- wordnet_show_106619065
|  |  |  | -- wordnet_broadcast_106619428
|  |  |  |  | -- wordnet_serial_106621447
|  |  |  |  | -- wordnet_television_program_106620579
|  |  |  |  | -- wordnet_game_show_106621061
|  |  |  | -- wordnet_play_107018931
|  |  |  |  | -- wordnet_musical_107019172
|  |  |  | -- wordnet_movie_106613686
|  |  |  |  | -- wordnet_musical_107019172
|  |  |  |  | -- wordnet_documentary_106616806
```

```
|  |  |  | -- wordnet_attraction_106615561
|  | -- wordnet_happening_107283608
|  |  | -- wordnet_incident_107307477
|  |  | -- wordnet_trouble_107289014
|  |  |  | -- wordnet_misfortune_107304852
|  |  |  |  | -- wordnet_mishap_107314427
|  |  |  |  |  | -- wordnet_accident_107301336
|  |  | -- wordnet_beginning_107290905
|  |  | -- wordnet_ending_107291312
|  | -- wordnet_group_action_101080366
|  |  | -- wordnet_vote_100182213
|  |  |  | -- wordnet_election_100181781
|  |  | -- wordnet_military_action_100952963
|  |  |  | -- wordnet_war_100973077
|  |  |  | -- wordnet_battle_100953559
|  |  |  |  | -- wordnet_naval_battle_100958477
|  |  | -- wordnet_social_control_101123598
|  |  | -- wordnet_conflict_100958896
| -- wordnet_organization_108008335
|  | -- wordnet_nongovernmental_organization_108009834
|  |  | -- wordnet_mission_108403225
|  |  | -- wordnet_denomination_108146782
|  | -- wordnet_unit_108189659
|  |  | -- wordnet_military_unit_108198398
|  |  |  | -- wordnet_army_unit_108190754
|  |  |  |  | -- wordnet_regiment_108213817
|  |  |  |  | -- wordnet_brigade_108213978
|  |  |  |  | -- wordnet_cavalry_108389710
|  |  |  |  |  | -- wordnet_squadron_108220089
|  |  |  |  | -- wordnet_division_108213205
|  |  |  |  | -- wordnet_battalion_108214083
|  |  | -- wordnet_family_108078020
|  |  | -- wordnet_administrative_unit_108077292
|  |  |  | -- wordnet_agency_108337324
|  |  |  |  | -- wordnet_law_enforcement_agency_108348815
|  |  |  | -- wordnet_council_108310949
|  |  |  | -- wordnet_intelligence_108339454
|  |  |  | -- wordnet_division_108220714
|  |  |  |  | -- wordnet_department_108114861
|  |  |  |  |  | -- wordnet_government_department_108119821
|  |  |  |  |  |  | -- wordnet_local_department_108120384
|  |  |  | -- wordnet_committee_108324514
```

```
|  |  | -- wordnet_team_108208560
|  |  |  | -- wordnet_baseball_team_108079319
|  |  |  | -- wordnet_hockey_team_108080386
|  |  |  | -- wordnet_basketball_team_108079852
|  |  |  | -- wordnet_football_team_108080025
|  |  | -- wordnet_political_unit_108359949
|  |  |  | -- wordnet_state_108168978
|  | -- wordnet_institution_108053576
|  |  | -- wordnet_company_108058098
|  |  |  | -- wordnet_broadcasting_company_108002015
|  |  |  | -- wordnet_subsidiary_company_108003935
|  |  |  | -- wordnet_electronics_company_108003035
|  |  |  | -- wordnet_food_company_108003427
|  |  |  | -- wordnet_service_108186047
|  |  |  |  | -- wordnet_utility_108185758
|  |  |  |  |  | -- wordnet_power_company_108186393
|  |  | -- wordnet_educational_institution_108276342
|  |  |  | -- wordnet_school_108276720
|  |  |  |  | -- wordnet_academy_108277805
|  |  |  |  | -- wordnet_graduate_school_108282696
|  |  |  |  |  | -- wordnet_business_school_108281812
|  |  |  |  | -- wordnet_private_school_108411170
|  |  |  |  |  | -- wordnet_seminary_108284994
|  |  |  |  |  | -- wordnet_boarding_school_108411701
|  |  |  |  | -- wordnet_secondary_school_108284481
|  |  |  |  |  | -- wordnet_senior_high_school_108409617
|  |  |  |  |  | -- wordnet_preparatory_school_108409969
|  |  |  |  |  | -- wordnet_comprehensive_school_108413248
|  |  |  |  | -- wordnet_grade_school_108412749
|  |  |  |  | -- wordnet_public_school_108410282
|  |  | -- wordnet_financial_institution_108054721
|  |  |  | -- wordnet_depository_financial_institution_108420278
|  |  |  | -- wordnet_foundation_108406486
|  |  |  |  | -- wordnet_charity_108406619
|  |  |  | -- wordnet_nondepository_financial_institution_108419984
|  |  |  |  | -- wordnet_insurance_company_108070465
|  |  | -- wordnet_religion_108081668
|  | -- wordnet_musical_organization_108246613
|  |  | -- wordnet_chorus_108187837
|  |  |  | -- wordnet_choir_108188638
|  |  | -- wordnet_dance_band_108249960
|  |  |  | -- wordnet_rock_group_108250501
```

```
|  |  | -- wordnet_ensemble_108188235
|  |  | -- wordnet_orchestra_108248157
|  | -- wordnet_enterprise_108056231
|  |  | -- wordnet_business_108061042
|  |  |  | -- wordnet_carrier_108057633
|  |  |  |  | -- wordnet_line_103671473
|  |  |  |  |  | -- wordnet_railway_104048568
|  |  |  |  |  | -- wordnet_airline_102690081
|  |  |  | -- wordnet_chain_108057816
|  |  |  |  | -- wordnet_restaurant_chain_108061801
|  |  |  | -- wordnet_manufacturer_108060446
|  |  |  | -- wordnet_firm_108059870
|  |  |  |  | -- wordnet_publisher_108062623
|  |  |  |  | -- wordnet_law_firm_108064039
|  |  |  |  | -- wordnet_corporation_108059412
|  |  |  | -- wordnet_agency_108057206
|  |  | -- wordnet_commercial_enterprise_108065093
|  |  |  | -- wordnet_cooperative_101100877
|  | -- wordnet_association_108049401
|  |  | -- wordnet_institute_108407330
|  |  | -- wordnet_league_108231184
|  |  |  | -- wordnet_football_league_108232496
|  |  | -- wordnet_club_108227214
|  |  |  | -- wordnet_golf_club_108229694
|  | -- wordnet_polity_108050385
|  |  | -- wordnet_government_108050678
|  | -- wordnet_party_108256968
|  | -- wordnet_force_108208016
|  |  | -- wordnet_military_service_108198137
|  |  |  | -- wordnet_army_108191230
|  | -- wordnet_company_108187033
|  | -- wordnet_deputation_108402442
|  |  | -- wordnet_diplomatic_mission_108402693
|  | -- wordnet_union_108233056
|  | -- wordnet_alliance_108293982
| -- wordnet_person_100007846
|  | -- wordnet_contestant_109613191
|  |  | -- wordnet_athlete_109820263
|  |  |  | -- wordnet_hockey_player_110179291
|  |  |  | -- wordnet_cricketer_109977326
|  |  |  | -- wordnet_football_player_110101634
|  |  |  | -- wordnet_ballplayer_109835506
```

```
|  |  |  | -- wordnet_basketball_player_109842047
|  |  |  |  | -- wordnet_forward_110105733
|  |  |  | -- wordnet_soccer_player_110618342
|  |  | -- wordnet_player_110439851
|  |  |  | -- wordnet_hockey_player_110179291
|  |  |  | -- wordnet_football_player_110101634
|  |  |  | -- wordnet_ballplayer_109835506
|  |  |  | -- wordnet_soccer_player_110618342
|  |  | -- wordnet_winner_110782940
|  |  |  | -- wordnet_medalist_110305062
|  | -- wordnet_peer_109626238
|  |  | -- wordnet_associate_109816771
|  |  |  | -- wordnet_colleague_109935990
|  | -- wordnet_intellectual_109621545
|  |  | -- wordnet_scholar_110557854
|  |  |  | -- wordnet_historian_110177150
|  |  |  | -- wordnet_alumnus_109786338
|  | -- wordnet_female_109619168
|  |  | -- wordnet_woman_110787470
|  | -- wordnet_communicator_109610660
|  |  | -- wordnet_negotiator_110351874
|  |  |  | -- wordnet_representative_110522035
|  |  |  |  | -- wordnet_head_of_state_110164747
|  |  |  |  |  | -- wordnet_sovereign_110628644
|  |  | -- wordnet_writer_110794014
|  |  |  | -- wordnet_poet_110444194
|  |  |  | -- wordnet_novelist_110363573
|  |  |  | -- wordnet_journalist_110224578
|  | -- wordnet_ruler_110541229
|  |  | -- wordnet_sovereign_110628644
|  | -- wordnet_adult_109605289
|  |  | -- wordnet_professional_110480253
|  |  |  | -- wordnet_educator_110045713
|  |  |  |  | -- wordnet_academician_109759069
|  |  |  | -- wordnet_lawyer_110249950
|  |  |  | -- wordnet_health_professional_110165109
|  |  |  |  | -- wordnet_medical_practitioner_110305802
|  |  |  |  |  | -- wordnet_doctor_110020890
|  |  | -- wordnet_woman_110787470
|  | -- wordnet_entertainer_109616922
|  |  | -- wordnet_performer_110415638
|  |  |  | -- wordnet_actor_109765278
```

```
|  |  |  | -- wordnet_musician_110340312
|  |  |  |  | -- wordnet_singer_110599806
|  | -- wordnet_leader_109623038
|  |  | -- wordnet_spiritual_leader_109505153
|  |  |  | -- wordnet_clergyman_109927451
|  |  |  |  | -- wordnet_priest_110470779
|  |  |  |  |  | -- wordnet_bishop_109857200
|  |  | -- wordnet_aristocrat_109807754
|  |  |  | -- wordnet_male_aristocrat_110285135
|  |  |  |  | -- wordnet_noble_110271677
|  |  | -- wordnet_politician_110451263
|  |  |  | -- wordnet_legislator_110253995
|  |  |  |  | -- wordnet_senator_110578471
|  |  |  | -- wordnet_mayor_110303814
|  |  | -- wordnet_head_110162991
|  |  |  | -- wordnet_administrator_109770949
|  |  |  |  | -- wordnet_director_110014939
|  |  |  |  | -- wordnet_executive_110069645
|  |  | -- wordnet_lawgiver_110249270
|  |  |  | -- wordnet_legislator_110253995
|  |  |  |  | -- wordnet_senator_110578471
|  |  | -- wordnet_trainer_110722575
|  |  |  | -- wordnet_coach_109931640
|  | -- wordnet_worker_109632518
|  |  | -- wordnet_skilled_worker_110605985
|  |  |  | -- wordnet_serviceman_110582746
|  |  |  |  | -- wordnet_military_officer_110317007
|  |  |  |  |  | -- wordnet_commissioned_officer_109942970
|  |  |  |  |  |  | -- wordnet_commissioned_military_officer_109943239
|  |  |  |  |  |  |  | -- wordnet_general_officer_110125786
|  |  |  |  |  |  |  |  | -- wordnet_general_110123844
|  |  |  | -- wordnet_official_110372373
|  |  |  |  | -- wordnet_diplomat_110013927
|  |  |  |  | -- wordnet_judge_110225219
|  | -- wordnet_scientist_110560637
|  | -- wordnet_creator_109614315
|  |  | -- wordnet_producer_110480018
|  |  |  | -- wordnet_film_maker_110088390
|  |  |  |  | -- wordnet_film_director_110088200
|  |  | -- wordnet_artist_109812338
|  |  |  | -- wordnet_musician_110339966
|  |  |  |  | -- wordnet_composer_109947232
```

```
|  |  |  | -- wordnet_painter_110391653
|  | -- wordnet_traveler_109629752
|  |  | -- wordnet_migrant_110314952
|  |  |  | -- wordnet_immigrant_110199489
|  |  | -- wordnet_absentee_109757653
|  |  |  | -- wordnet_exile_110071332
|  | -- wordnet_disputant_109615465
|  |  | -- wordnet_reformer_110515194
|  |  |  | -- wordnet_militant_110315837
|  | -- wordnet_preserver_110466918
|  |  | -- wordnet_defender_109614684
|  | -- wordnet_unfortunate_109630641
|  | -- wordnet_expert_109617867
|  | -- wordnet_adjudicator_109769636
|  |  | -- wordnet_judge_110225219
|  | -- wordnet_good_person_110138767
|  | -- wordnet_authority_109824609
|  |  | -- wordnet_civil_authority_110541833
|  |  |  | -- wordnet_mayor_110303814
|  | -- wordnet_combatant_109939313
| -- yagoGeoEntity
|  | -- wordnet_location_100027167
|  |  | -- wordnet_region_108630985
|  |  |  | -- wordnet_geographical_area_108574314
|  |  |  |  | -- wordnet_tract_108673395
|  |  |  |  |  | -- wordnet_plot_108674739
|  |  |  |  |  |  | -- wordnet_garden_103417345
|  |  |  |  |  | -- wordnet_site_108651247
|  |  |  |  |  |  | -- wordnet_cemetery_108521623
|  |  |  |  |  | -- wordnet_park_108615374
|  |  |  |  |  | -- wordnet_park_108615149
|  |  |  |  |  |  | -- wordnet_national_park_108600992
|  |  |  |  |  | -- wordnet_subdivision_108674251
|  |  |  |  | -- wordnet_urban_area_108675967
|  |  |  |  |  | -- wordnet_municipality_108626283
|  |  |  |  |  |  | -- wordnet_town_108665504
|  |  |  |  |  |  |  | -- wordnet_ghost_town_108671509
|  |  |  |  |  |  | -- wordnet_city_108524735
|  |  |  |  | -- wordnet_settlement_108672562
|  |  |  |  |  | -- wordnet_village_108672738
|  |  |  | -- wordnet_area_108497294
|  |  |  |  | -- wordnet_scene_108645963
```

```
|  |  |  |  |  | -- wordnet_venue_108677628
|  |  |  |  | -- wordnet_section_108648322
|  |  |  |  |  | -- wordnet_vicinity_108641113
|  |  |  |  | -- wordnet_center_108523483
|  |  |  |  |  | -- wordnet_seat_108647945
|  |  |  |  |  |  | -- wordnet_county_seat_108547143
|  |  |  | -- wordnet_district_108552138
|  |  |  |  | -- wordnet_administrative_district_108491826
|  |  |  |  |  | -- wordnet_state_108654360
|  |  |  |  |  | -- wordnet_municipality_108626283
|  |  |  |  |  |  | -- wordnet_town_108665504
|  |  |  |  |  |  |  | -- wordnet_ghost_town_108671509
|  |  |  |  |  |  | -- wordnet_city_108524735
|  |  |  |  |  | -- wordnet_country_108544813
|  |  |  |  |  | -- wordnet_commune_108541609
|  |  |  |  |  | -- wordnet_township_108672199
|  |  |  |  |  | -- wordnet_borough_108540532
|  |  |  |  |  | -- wordnet_school_district_108587709
|  |  |  |  | -- wordnet_residential_district_108553535
|  |  |  |  |  | -- wordnet_suburb_108554440
|  |  | -- wordnet_space_113910384
|  |  |  | -- wordnet_opening_109379111
|  |  |  |  | -- wordnet_crack_109258715
|  |  |  |  |  | -- wordnet_vent_109470550
|  |  | -- wordnet_point_108620061
|  |  |  | -- wordnet_position_108621598
|  |  |  |  | -- wordnet_landmark_108624891
|  |  |  | -- wordnet_topographic_point_108664443
|  |  |  | -- wordnet_geographic_point_108578706
|  |  |  |  | -- wordnet_address_108491027
|  |  |  |  |  | -- wordnet_residence_108558963
|  |  |  |  |  |  | -- wordnet_home_108559508
|  |  |  |  | -- wordnet_workplace_104602044
|  |  | -- wordnet_region_108630039
|  |  |  | -- wordnet_county_108546183
|  |  |  | -- wordnet_extremity_108568978
|  |  | -- wordnet_line_108593262
|  |  |  | -- wordnet_path_108616311
|  | -- wordnet_structure_104341686
|  |  | -- wordnet_building_102913152
|  |  |  | -- wordnet_place_of_worship_103953416
|  |  |  |  | -- wordnet_church_103028079
```

```
|  |  |  |  | -- wordnet_temple_104407435
|  |  |  | -- wordnet_skyscraper_104233124
|  |  |  | -- wordnet_house_103544360
|  |  |  |  | -- wordnet_residence_104079244
|  |  |  |  |  | -- wordnet_religious_residence_104073948
|  |  |  |  |  |  | -- wordnet_monastery_103781244
|  |  |  |  | -- wordnet_mansion_103719053
|  |  |  |  |  | -- wordnet_palace_103878066
|  |  |  | -- wordnet_medical_building_103739518
|  |  |  |  | -- wordnet_hospital_103540595
|  |  |  | -- wordnet_hotel_103542333
|  |  |  | -- wordnet_theater_104417809
|  |  | -- wordnet_memorial_103743902
|  |  | -- wordnet_housing_103546340
|  |  |  | -- wordnet_dwelling_103259505
|  |  |  |  | -- wordnet_house_103544360
|  |  |  |  |  | -- wordnet_residence_104079244
|  |  |  |  |  |  | -- wordnet_religious_residence_104073948
|  |  |  |  |  |  |  | -- wordnet_monastery_103781244
|  |  |  |  |  | -- wordnet_mansion_103719053
|  |  |  |  |  |  | -- wordnet_palace_103878066
|  |  | -- wordnet_building_complex_102914991
|  |  |  | -- wordnet_plant_103956922
|  |  | -- wordnet_establishment_103297735
|  |  |  | -- wordnet_place_of_business_103953020
|  |  |  |  | -- wordnet_mercantile_establishment_103748162
|  |  |  |  |  | -- wordnet_plaza_103965456
|  |  |  | -- wordnet_institution_103574555
|  |  |  |  | -- wordnet_penal_institution_103907654
|  |  |  |  |  | -- wordnet_correctional_institution_103111690
|  |  | -- wordnet_stadium_104295881
|  |  | -- wordnet_obstruction_103839993
|  |  |  | -- wordnet_barrier_102796623
|  |  |  |  | -- wordnet_dam_103160309
|  |  | -- wordnet_area_102735688
|  |  |  | -- wordnet_room_104105893
|  |  |  |  | -- wordnet_library_103660909
|  |  | -- wordnet_tower_104460130
|  |  | -- wordnet_bridge_102898711
|  | -- wordnet_facility_103315023
|  |  | -- wordnet_station_104306080
|  |  |  | -- wordnet_terminal_104412901
```

```
|  |  |  |  | -- wordnet_railway_station_104049098
|  |  |  | -- wordnet_broadcasting_station_102903405
|  |  |  |  | -- wordnet_radio_station_104044119
|  |  |  |  | -- wordnet_television_station_104406350
|  |  |  |  |  | -- wordnet_channel_103006398
|  |  |  | -- wordnet_power_station_103996655
|  |  | -- wordnet_military_installation_103763133
|  |  |  | -- wordnet_military_post_103763403
|  |  |  |  | -- wordnet_garrison_103420559
|  |  | -- wordnet_depository_103177349
|  |  |  | -- wordnet_museum_103800563
|  |  | -- wordnet_airfield_102687992
|  |  |  | -- wordnet_airport_102692232
|  | -- wordnet_body_of_water_109225146
|  |  | -- wordnet_stream_109448361
|  |  |  | -- wordnet_river_109411430
|  |  | -- wordnet_lake_109328904
|  |  | -- wordnet_bay_109215664
|  | -- wordnet_geological_formation_109287968
|  |  | -- wordnet_beach_109217230
|  |  | -- wordnet_natural_depression_109366017
|  |  |  | -- wordnet_valley_109468604
|  |  | -- wordnet_natural_elevation_109366317
|  |  |  | -- wordnet_mountain_109359803
|  |  |  | -- wordnet_hill_109303008
|  |  | -- wordnet_volcanic_crater_109472413
|  |  | -- wordnet_range_109403734
|  | -- wordnet_way_104564698
|  |  | -- wordnet_road_104096066
|  |  |  | -- wordnet_highway_103519981
|  |  |  |  | -- wordnet_expressway_103306610
|  |  |  | -- wordnet_thoroughfare_104426618
|  |  |  |  | -- wordnet_street_104334599
|  |  | -- wordnet_passage_103895293
|  | -- wordnet_land_109334396
|  |  | -- wordnet_island_109316454
```