AUTOMATED CURATION OF A DATASET FOR NAMED ENTITY RECOGNITION FROM Wikipedia

by

Furkan Enes Yalçın & Nazmican Çalık

Submitted to the Department of Computer
Engineering in partial fulfillment of
the requirements for the degree of
Bachelor of Science

Undergraduate Program in Computer Engineering
Boğaziçi University

Spring 2019

AUTOMATED CURATION OF A DATASET FOR NAMED ENTITY RECOGNITION FROM Wikipedia

APPROVED BY:

Dr. Suzan Üsküdarlı .....
(Project Supervisor)

DATE OF APPROVAL:

ACKNOWLEDGEMENTS

ABSTRACT

AUTOMATED CURATION OF A DATASET FOR NAMED ENTITY RECOGNITION FROM Wikipedia

Most NER datasets do not contain many lexically ambiguous words, as a result it becomes hard for a NER model trained with such a dataset to distinguish between different meanings of a lexically ambiguous word. For example, in the sentence "Paris Hilton owns a Hilton Hotel in Paris.", Both Paris and Hilton are used with two different meanings and their entity types are different too. (Paris: Person, Paris: Location, etc.) In order to deal with this problem, we decided to come up with a method to curate a dataset that intensely contains such words, with their entity types in a language independent manner. We decided to use Wikipedia as our data source because of the number of available languages and its potential for such words. These words are brought together under the category called "Disambiguation pages". We get potential meanings of an ambiguous term from these pages and then extract sentences from pages that refers to these potential meanings. We call these potential meanings "Disambiguation Term Candidate (DT)". Later, we extract the entity type information of a DT from yet another Wikimedia Site, Wikidata. After we fetch all of these information we gather them all and create a useful data set for NER.

ÖZET

ADLANDIRILMIŞ VARLIK TANIMADA KULLANILMASI AMACIYLA VİKİPEDİ'DEN OTOMATİK OLARAK VERİ SETİ OLUŞTURMA

Çoğu NER veriseti fazla eşsesli söz öbeği içermiyor, bu sebeple bu verisetleriyle eğitilen NER modelleri eşsesli söz öbeklerinin farklı anlamlarını birbirinden ayırmakta zorlanıyor. Örneğin, "Paris'teki Hilton Oteli Paris Hilton'undur." cümlesinde hem Paris hem de Hilton kelimeleri iki farklı anlamda kullanılmıştır, ayrıca bu anlamların varlık tipleri de farklıdır. (Paris: İnsan, Paris. Mekan, v.b.) Bu sorunla başa çıkmak amacıyla içinde bir çok eşsezli söz öbeği barındıran verisetleri oluşturmaya yönelik ve herhangi bir dil için çalışabilecek bir metod geliştirmeye karar verdik. Böyle bir çok söz öbeği barındırdığını düşündüğümüz ve birden fazla dilde metinlere sahip olduğu için Vikipedi'yi kaynak olarak seçtik. Vikipedi bünyesinde bu kelimeler "Anlam ayrımı" kategorisi altında toplanmıştır. Bu çok anlamlı sözlerin olası anlamlarını bu sayfalardan elde edip, sonrasında bu sayfalar aracılığıyla olası anlamların içinde geçtiği cümleleri de elde ediyoruz. Sonrasında, bu olası anlamların varlık tiplerini de bir başka Wikimedia sitesi olan Wikidata aracılığıyla öğreniyoruz. Tüm bu veriyi çektikten sonra bunları birleştirip kullanışlı bir veriseti elde ediyoruz.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

LIST OF ACRONYMS/ABBREVIATIONS

AT	Ambiguous Term - e.g. Beşiktaş
DT	Disambiguated Term - e.g. e.g. Beşiktaş Football Team, Beşiktaş (City)
VDT	Valid Disambiguated Term. Selected from DTs. - e.g. Beşiktaş Football Team, Beşiktaş (City)
ETG	Entity Type Graph - Graph of classes that the entity belongs to. In our case, entities are VDTs.
WP	Wikipedia Page
SE	Sentence with entity. <Sentence, beginning index of entity, stopping index of entity>
TAG	Tag of the disambiguation term in a sentence - e.g. Location.
PER	Person
ORG	Organization
LOC	Location
ORGLOC	Organization-Location

1. INTRODUCTION AND MOTIVATION

Named Entity is a real world object such as persons, locations, organizations, products, etc., that can be denoted with a proper name. For example, Charlie Chaplin which is a famous person, is an entity in a text. Further examples include Eiffel Tower, World Health Organization etc. However, terms that don't refer to the instances of real world objects are not recognized as named entities. For instance, car is not a named entity because it is too generic and don't refer to an instance of an entity.

In information science, one of the most challenging tasks is to recognize which named entity belongs to which entity type. Type of entities include locations, persons, organizations, etc. Recognizing unique named entities is a problem and if the surface form of the named entity is ambiguous then the problem is harder. In the following example we will try to illustrate the problem.

Let's say our sentence is "Paris Hilton geçen ay yeni bir otel aldığını duyurdu." In this sentence Paris and Hilton words are named entities. The problem is to decide if the word "Paris" refers to the city Paris or the name of the famous Paris Hilton. Another instance of the problem in this sentence is Hilton. It could refer to the surname of the Paris Hilton or the Hilton Hotels which is an organization. In order to develop a good statistical model for the problem there is a need for a good dataset. Our aim is to curate a dataset that is including ambiguous terms and their disambiguation term candidates in a sentence for a supervised machine learning model.

Yet another motivation for us is the lack of supervised datasets in alternative languages. [1]. Datasets created for NER tasks are typically curated for specific languages.(e.g.Portegese [2] With the methods we are proposing, it will be possible to construct a dataset for such languages.

2. STATE OF THE ART

There are many datasets available that are created for sake of NER algorithms. However, in most of these datasets there are only a handful of surface forms that are disambiguous, which means tagged with multiple entity types - e.g. "Paris: Location" and "Paris: Person". We predict that NER models, trained with such datasets can't perform well when they encounter these words. Since they did not encounter these words with their different meanings on training set, they cannot distinguish between a Person Paris and a Location Paris. But NER model do not encounter with these kind of words in the test sets either because the whole dataset is lacking these words which has the same surface form but are different entitites.

We hope to achieve to improve the performances of such models by providing them with a dataset which includes a high proportion of such words. In fact, datasets created with our method will mainly consist of such words.

3. METHODS

In this project we will try to get as many sentence as possible from Wikipedia Turkish pages to create a dataset. Our final aim is to create an automated tool that scraps the Turkish Wikipedia disambiguation pages and collects sentences.

Before we move on to the models that we used when solving the problem, we have to introduce some terminology.

- **(AT) Ambiguous Term:** Ambiguous Term is the word or words, that we are trying to determine its entity type. These terms may have several meanings. For example, Fenerbahçe which can mean the district or the Football Team. We get these terms from the disambiguation pages in Wikipedia. Every entry in those pages, leads to several meanings, semantics which have their own Wikipedia pages.

- **(DT) Disambiguated Term:** These are the terms that carry a specific meaning in a disambiguation page of an AT. Each DT offers a different semantic for the AT. A good dataset should include as big sets of sentences as possible for as many different DT as possible for an AT.

- **(VDT) Valid Disambiguated Term:** These are the disambiguation terms that are actually useful for the dataset. They are valid disambiguation terms because they include the ambiguation term in their surface form. This validation method is configurable.

- **(ETG) Entity Type Graph:** This is a graph of entity types. We gather these classes from Wikidata.

- **(ET) Entity Type:** This refers to the type of an entity. In ETG.

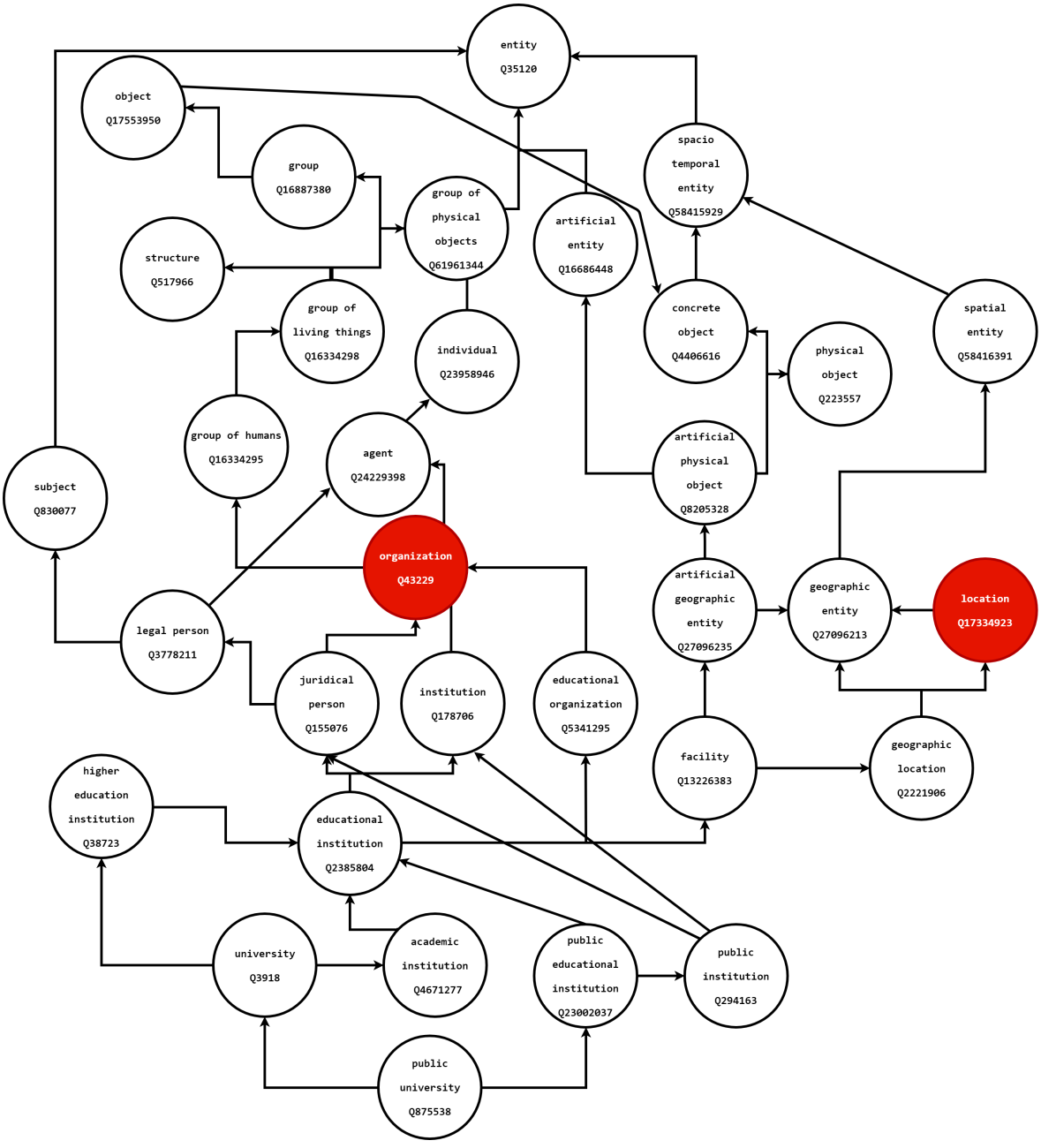

Figure 3.1. ETG of Boğaziçi University.

- **Tag:** This is the tag of a data entry in the dataset. This can be one of the following predetermined values: Location, Person, Organization, Organization-Location etc.

Our final goal is to curate a dataset that consist of tuples in the following format:

<AT, VDT, SE, TAG>

Let's see how we construct these data step by step.

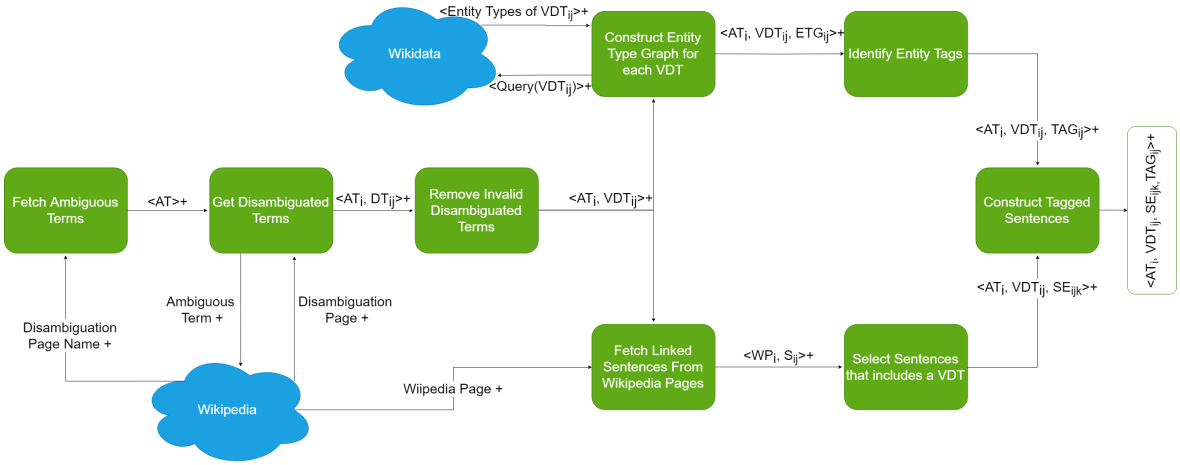

Figure 3.2. Methodology.

(i) **Fetching Ambiguous Terms**

In this first step our aim is to get all the ambiguous terms in following format.

<AT>+

For this we used the python pywikibot and crawled the page for dismabiguation pages list. That way we get all the ambiguous terms in the Wikipedia database.

(ii) **Getting Disambiguated Terms**

In this step our aim is to get all disambiguation terms for each AT, in following format.

<AT, DT+>+

For this, we crawled the dismabiguation page of every AT. From these pages, we extract all links to other Wikipedia pages and assume that they disambiguate the AT.

(iii) **Remove Invalid Disambiguated Terms**

In this third step, our aim is to filter out the unnecessary disambiguation terms. Some of these were irrelevant to our ambiguous term. For example in every case there is a disambiguated term page linked to general disambiguation page. The data format is not so different than the first step. This time we have the VDT's instead of DT's.

<AT, VDT+>+

We applied a simple logic when determining if a disambiguated term is valid or not. If the ambiguous term is a substring of the disambiguated term candidate, then we concluded that this DT is a VDT. The reasoning behind this was the following, if the text included the ambiguous term only then we will try to get the meaning of it.

Important Note: After this step we divided the process into two subprocesses. One process is responsible for finding the ETG (Entity Type Graph) and the other gets the text from the referenced pages of the DT's.

(iv) **Constructing the ETG For Every AT, VDT Pair**

In this step we find the ETG of a <AT,VDT> pair. For this purpose we have used Wikidata. When we are finished with this step we want to have the following data points for every <AT,VDT> pair. We represent the graph in GraphML format.

<AT, VDT, ETG>+

For each valid disambiguated term, we query the Wikidata entry. Since most pages in Wikipedia is mapped to Wikidata, we can get their Wikidata entry via their Wikipedia page. Then we first get the "instance of" field of the VDT, by doing this we find the class of the VDT. Then, we query the "subclass of" field of this class. We add these newly found classes to the graph and add edges from former class to the these newly found classes. We continue this operation for each class in the ETG until there is no new class is found. Figure 3.1 shows an example ETG.

(v) **Identifying Entity TAGs for each AT, VDT Pair**

In this step we identify the entity type of each VDT using its ETG.

<AT, VDT, TAG>+

We search the nodes of the ETG for predetermined entity types. These predetermined entity types are person, organization, location and can be changed. If the ETG contains one or many of these we tag the VDT. For example, if ETG contains "organiation" class, current TAG becomes "ORG". If it contains both "organization" and "location" current TAG becomes "ORGLOC".

(vi) **Getting The Sentences From Wikipedia Dump**

We were normally planning to get the pages and pages text from live Wikipedia page. However we have realized that it will pose time related problems. The reason is the following, making a network call to curl the Wikipedia page introduces a network lag. Considering there are more than twelve thousand pages, it is not affordable. That's why we have decided to use the Wikipedia pages dump. This dump is in XML format. There are several different dumps for Wikipedia. Some of them gives metadata on the other hand some gives the pages information. The dump we have worked with includes all Turkish Pages on Wikipedia.

In this step we find the sentences that makes the barebones of our dataset. We

try to get the sentences that includes links to vdt's. Detect the location of vdt's in the text and sace the start and end index as well. We put the sentence and the pair into the following format.

<AT, VDT, SE>+

Parsing a big XML file was harder then we expected. We needed to come up with a way to parse the Wikipedia dump efficiently as it is a big file. We have found a Python package called mwxml that is created for the purpose of parsing the Wikipedia dumps. We have parsed the Wikipedia dump with mwxml, it creates a generator object that traverses the pages in the dump.
For each page that is not a disambiguation page, the following process is applied. Links in the Wikipedia page follows the following form.

- [[Link|Seen Text]]
- [[Link]]

With the help of regex we capture the links in the page. After capturing the links, we hash the links (whole link) and save the hash value into a dictionary. Then, we get the salt text of the page with the help of a python package called mwparserfromhell.
Then we seperate the sentences via nltk. We then filter the sentences that includes templates, comments etc. Then we traverse the hash map that has the keys and replace the links with their seen texts and save the sentence with the final format.

<AT, VDT, SE>+

(vii) **Constructing The Final Form of The Dataset**

In this step our aim is to finalize the dataset using all the information we got until now. We just needed to merge the TAG and the Sentences for every <AT, VDT, SE>+ tuple. The final data format is in CoNNL-U format which is an

extended version of CoNNL-X. [3].For every sentence we separate the words and tag the entities, which allows us to convert the dataset to CoNNL-U format.

That is our way to solve the problem and construct a dataset for NER Models that is rich in terms of ambiguous terms.

4. RESULTS

We ran our method for Turkish Wikipedia and constructed a dataset to prove that our method works. This dataset contains 213,974 sentences with 247,191 tagged entities which is a relatively big number for a dataset. Some of the sentences may include noise, however most of the sentences are clean and suitable for the dataset. Even if we lost almost half of the entity tags, tool has constructed a dataset that is relatively big.

We validated that the proposed approach can be applied for any languages which Wikipedia supports. Furthermore, the dataset can be continuously updated with up-to-date entity references, as Wikipedia pages are also continuously updated.

Sample outputs of intermediate stages:

- <AT, VDT, SE, START, END>
  - “beşiktaş”, “Beşiktaş (basketbol takımı)”, "Beşiktaş, kurulduğu 1933 yılından bu yana Türkiye Basketbol Ligi’nde toplam 2 defa şampiyon olmuştur.", 0, 8
  - “tünel”, ”Tünel (semt)”, "Adını, Pera da denen, Tünel-Taksim arasında uzanan İstiklal Caddesi ve ona açılan sokakların belirlediği alanı kapsayan Beyoğlu semtinden alır.", 22, 27
  - “new jersey”, “New Jersey (albüm)”, ”Slippery When Wet ve ardından 1988 yılında yayınlanan New Jersey albümleri ile grup dünya çapında büyük bir üne erişti.” ,54 , 64
- <AT, VDT, TAG>
  - “beşiktaş”, “Beşiktaş (basketbol takımı)”, “ORG”
  - “beşiktaş”, "Beşiktaş, Beşiktaş", “LOC”
  - “sabancı”, ”Sabancı Üniversitesi”, “ORGLOC”
  - “sabancı”, “Güler Sabancı”, “PER”

5. CONCLUSION AND DISCUSSION

After all, the proposed system works. We were successful collecting sentences from Wikipedia. The dataset we created with the tool used Turkish Wikipedia Pages from the wikipedia pages dump. We successfully collected 213,974 sentences in total. Some of them include more than one entity so that we have extracted 247,191 named entities in total for Turkish language. If we improve our strategies, we can be able to increase this number.

Secondly, the tool we created curates datasets that are up to date. Wikipedia is a community sourced platform, the information (pages, links etc) get updated every day. As new pages are introduced, the system will find the current, up to date sentences. Whenever the users wish they can update their datasets by running the tool with the new dump.

We created a program that is language independent. That means as long as the Wikipedia pages exist, we can change the parameter of the program and collect data for other languages. We have tried to write the code for our program as flexible as possible. Since the full execution of the program takes a relatively long time, we have split the execution to several steps. We record every data we get after each step. That way if the execution stops at a point we don't lose any data.

With improvements to the existing system we can increase the number of sentences sharply. We showed that the proposed system works and language independency is a huge plus. System can be used for the languages that are poor in terms of NER datasets.

6. FUTURE WORK

We are simply proposing a method to curate a dataset for NER but we are also writing a code for this so that we can evaluate its usefulness. We divided our algorithm to several steps. Each step can be improved, perfected or customized separately. The followings are the possible improvements:

- **Improving the sentence extraction:** For now we are using nltk tools without any configuration. Some of the sentences are not separated correct with this tool but adjustments are possible to tune nltk to extract sentences with a better accuracy.

- **Increasing the number of sentences that are taken from Wikipedia pages:** For now we are only taking the sentences that are giving link to the corresponding VDT. What we can do in the future is to provide a logic or heuristic that will allow us to take the sentences that don't give any links, however, that are useful.

- **Creating a user interface to provide statistical data on curation process:** For now we don't have an interface to show the statistics of data crawling process. In the future we can create a web based solution to see the data flow from Wikipedia to our dataset. It can be helpful to see the success rate of the dataset curating process. It might allow us to intervene the process as humans if we detect any mistake that the program does e.g. Adding a sentence that does not actually includes the AT.

So, all of these can be used just as a guideline to curate a useful dataset for NER Models, or people can directly use our program too. We are currently using Wikipedia as our data source. There may exist other data sources which are more suitable for this purpose or can be created in the future solely for this purpose.We believe that usage

of datasets created by our method will greatly improve an NER model's accuracy on texts that intensely contain lexically ambiguous words.

REFERENCES

1. Lample, G., M. Ballesteros, S. Subramanian, K. Kawakami and C. Dyer, "Neural architectures for named entity recognition", *arXiv preprint arXiv:1603.01360*, 2016.

2. Santos, D., N. Seco, N. Cardoso and R. Vilela, "Harem: An advanced ner evaluation contest for portuguese", *quot; In Nicoletta Calzolari; Khalid Choukri; Aldo Gangemi; Bente Maegaard; Joseph Mariani; Jan Odjik; Daniel Tapias (ed) Proceedings of the 5 th International Conference on Language Resources and Evaluation (LREC'2006)(Genoa Italy 22-28 May 2006)*, 2006.

3. Buchholz, S. and E. Marsi, "CoNLL-X shared task on multilingual dependency parsing", *Proceedings of the tenth conference on computational natural language learning*, pp. 149–164, Association for Computational Linguistics, 2006.

APPENDIX A: DATA AVAILABILITY STATEMENT

APPENDIX B: STANDARDS, LAWS, REGULATIONS AND DIRECTIVES