International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 11, No.1 (Feb-2022)

https://dx.doi.org/10.12785/ijcds/110161

ClassifyWiki: An Experimental Study on Building Generic-Type Wikipedia Classifiers

Michel Naim Gerguis[1], M. Watheq El-Kharashi[2] and Cherif Salama[3,2]

[1]*Microsoft, Cairo, Egypt*
[2]*Department of Computer and Systems Engineering, Faculty of Engineering, Ain Shams University, Cairo, Egypt*
[3]*The American University in Cairo, Cairo, Egypt. Ain Shams University, Cairo, Egypt*

Received 26 Aug. 2021, Revised 28 Jan. 2022, Accepted 1 Feb. 2022, Published 15 Feb. 2022

Abstract: This paper introduces ClassifyWiki, a framework that automatically generates Wikipedia-based text classifiers using a small set of positive training articles. ClassifyWiki aims to simplify the process of collecting hundreds or thousands of Wikipedia pages with the same entity class, using a set of positive articles with sizes possibly as small as 10 pages. The customer could define the small initial set at any different level of granularity (i.e. people, sports people, or even footballers). ClassifyWiki leveraged many previous efforts in Wikipedia entity classification in order to build a generic framework that works for any entity at any level of granularity with few examples. The framework does not only offer a set of pre-built models for many entity classes but a tool tuned through hundreds of experiments to generate models for any given set of articles. To test the framework, we manually tagged a data set of 2500 Wikipedia pages. This data set covers 808 unique entity classes on different levels of granularity. ClassifyWiki was tested over 103 different entity classes varying in size down to only 5 positive articles. On our blind set, ClassifyWiki achieved a macro-averaged f1-score of 83% with 96% precision and 74% recall using 50 or more positive articles.

Keywords: ClassifyWiki, Entity Classification, Fine-Grained Entity Classification, Text Classification, Wikipedia Classification.

1. Introduction

Machine understanding attracted lots of researchers in the last decade. Extracting the types, entity classes, of any text is a main task for machine understanding. Hoffart et al. [1] simply state it as converting strings to things. Fine grained classification is urgently needed to support a better experience in machine understanding [2]. Wikipedia is considered a great source for many knowledge extraction systems from free text [3], [4] and from its rich structured and semi-structured data [5], [6]. Wikipedia hyperlinks inspired lots of named entity extractors [2], [7].

In this paper, we present ClassifyWiki, a framework that can automatically build a reliable binary classifier for any input set of Wikipedia articles. This set can be of any level of granularity. For example, it can be a set of pages about female singers, rock albums, Spanish football clubs, populated places, or even natural disasters. The smaller the set size is, the better for the real world scenarios. We defined the problem as entity classification problem as we mainly target entities not generic concepts. We're motivated to build a framework that is not bounded by specific set of entity classes like previous efforts. Generating a reliable classifier for any entity class in any level of granularity is the goal of this framework. ClassifyWiki dose not target head entity classes only but tail ones that is why we tuned ClassifyWiki to the best marco-averaged f1-scores. Currently, many knowledge bases (KBs) like Dbpedia contain precise structured information, ClassifyWiki could be questionable. Although precise information could be easily grasped from KBs due to the human intervention, they suffer from the limited coverage and probably outdated or incomplete information. Having such a lightweight framework that runs over any Wikipedia dump using a small set of positive articles could help bridging this gap.

Lots of applications could emerge using ClassifyWiki. [1] introduced a new flavor for Culturomics [8] by shifting the analytics direction from spotting trends of strings to things, entities, and also categories from news articles. ClassifyWiki adds a new experience for the Culturomics projects by ingesting lots of fine grained classifiers to track any type of interest. This framework is a major building block in an culturomics-like analytics framework for Wikipedia content [9].

Although ClassifyWiki is about entity classification (song, novel, or football player) not topic modeling (sports,

E-mail address: mgerguis@microsoft.com, watheq.elkharashi@eng.asu.edu.eg, cherif.salama@aucegypt.edu

finance, or entertainment), it can be easily experimented to model topics. Every Wikipedia page represents the history of an entity or a concept which is not the case in web/text documents. To model sports, ClassifyWiki could be positioned to generate classifiers for sports people, clubs, and stadiums collecting lots of articles starting from few examples. Starting from tens of pages, ClassifyWiki would collect a reasonable size of training data for a generic text classifier to model sports using non-Wikipedia specific features such as text grams. Hence, ClassifyWiki can reduce the manual tagging effort needed as long as the modeled topic is covered by Wikipedia pages like [10]. The same experience could be extended to model topics in many languages as ClassifyWiki is a language independent framework. Using few English seeds, ClassifyWiki would generate many articles in the same entity classes which can be mapped using Wikipedia language links to the desired language. Then, the framework could build a new classifier using the mapped articles.

ClassifyWiki could also, help in fine-grained Named Entity Recognizers (NERs). Many NERs are built using Wikipedia hyperlinks after classifying Wikipedia articles. ClassifyWiki could step in to build NERs for fine-grained entity classes or any user defined ones that will be very useful in information extraction tasks [2], [11].

Our main contributions could be summarized as follows:

- ClassifyWiki, a framework to build generic-type Wikipedia classifiers. ClassifyWiki automates the process of generating Wikipedia classifiers out of small-sized sets of positive articles. The input set can potentially have as little as 10 pages (Sections 5, 6).
- A survey of Wikipedia-based classification features suggested by the literature along with an analysis of the dominance of each one across the whole Wikipedia dump (Section 4).
- A detailed analysis and comparison of the contribution of every feature (Sections 5-B, 6-C) along with an experimentation study using forward feature selection. This feature set targets reducing the dimensionality in order to support the small number of training instances (Sections 5-C, 6-D).
- A new data set of 2500 randomly selected and labeled Wikipedia articles. Whenever applicable, each page was manually tagged with multiple labels of various granularity levels. In total the data set covers 808 fine-grained classes (Section 3).

ClassifyWiki's implementation and the data set are made available for the research community upon request.

2. Related Work

In this section we briefly summarize some efforts related to Wikipedia articles classification and compare our framework against them. ClassifyWiki cannot be directly compared with previous Wikipedia classifiers since it builds a new generic layer that complements them. Most of the previous efforts targeted specific classes (person, location, and organization) and tuned their systems consequently whereas ClassifyWiki provide a framework to generate classifiers for any given set of pages.

In previous efforts, mainly heuristic rules were adopted to classify Wikipedia [12], [13], [14]. DBpedia [15] and Freebase [16] are the largest well-known sources of structured data, knowledge bases (KBs), out of Wikipedia based on heuristics. The patterns mainly relied on infoboxes, coordinate templates, and keywords derived from the first sentences or the categories. Although the patterns showed good results in terms of precision, the recall was an issue due to their limited coverage. YAGO ontology [17], [18] was introduced using heuristics on Wikipedia structured and semi-structured resources. YAGO classifies the pages relying on Wikipedia's categorization scheme. Heuristics were used in extracting the page type out of its categories using the stem of the plural head noun of the category. It cannot collect Wikipedia pages for any user-defined entity class as it is bounded by Wikipedia categorization. Heuristic patterns are not scalable. That is why building a classifier for a new entity class would require a new list of patterns. ClassifyWiki is not about heuristic patterns in order to scale. The framework can build a classifier for any user-defined set of Wikipedia articles whether it is a coarse-grained entity class, a fine-grained entity class, or even a different concept.

A novel approach was introduced to generate NER training data out of Wikipedia using hyperlinks after classifying the articles [7]. A heuristic-based bootstrapping approach to classify Wikipedia was adopted depending mainly on category type and definition sentence type heuristics [19], [17]. Bhole et al. [12] classified Wikipedia pages using the page tokens in a pre-processing step in their dynamic timeline generation approach. Also, Kazama and Torisawa [19] experimented leveraging the types of Wikipedia pages as features in NER. They introduced deducing the page type from the head noun of the first sentence after a copula, is, was, are, or were. Dakka and Cucerzan [20] introduced two-views concept for any page (local and global). A local view in which the features are derived from the page itself and a global one inspired by Wikipedia hyperlinks, the incoming links for any page from another one. The first paragraph was a major source for features used in previous efforts.

Lots of efforts tackled the problem as a text classification problem adopting support vector machine (SVM) based classifiers. Most of the efforts were directed mainly to 3 classes: persons, locations, and organizations. A miscellaneous class was considered in some previous efforts in order to collect entities that do not fit in the previous 3 classes and improve the precision of the overall system. ClassifyWiki leveraged those previous systems. Although they tackled the classification process of few top entity classes, the huge

effort in feature engineering is valuable for ClassifyWiki.

Targeting non-English Wikipedia classifiers [21] and language independent feature sets was the evolution of this problem in order to use the same classifier in any Wikipedia language [22]. The problem then, extended to target fine-grained classification. Tkachenko et al. [23] tackled 15 entity classes. ClassifyWiki was inspired by the idea to enable fine-grained and user-defined entity classes as well. ClassifyWiki tackled the problem of building a framework, which can generate a classifier for any input Wikipedia pages not to generate a model for some predefined classes in any language.

Most of the previous systems collected and labeled their own dataset. No unified dataset was used in the task. Some systems used random sampling out of the whole dump. Other systems used heuristics after labeling a random sample whether to automatically tag more pages using Wikipedia list pages [20] or to select popular pages to be manually tagged using incoming links count and existance of the page in many langauges as popularity measures [7], [24]. Collecting lists from web pages given a sample for each class was another selection process that was used [23]. Some concerns were raised against random sampling in the dataset selection process in order not to miss important types like countries in location class. We were in favor of random sampling technique to represent the whole spectrum.

The number of training examples considered in previous efforts is the range of thousands of articles. ClassifyWiki aimed to minimize the number of training examples. We experimented with different sizes as little as only five examples for each entity class.

We can summarize the novelty of ClassifyWiki framework against previous similar efforts as follows:

- Offering hundreds of pre-built classifiers for all entity classes in ClassifyWiki dataset.
- The ability to classify coarse-grained, fine-grained, and user-defined entity classes.
- The considered number of training articles as little as few tens of examples.

3. Data Set

In this section, we present the data set we built. We also explain why we had to construct one, how we constructed it, including the challenges we encountered, and finally we give some interesting insights about it. A 2015 Wikipedia English XML dump was used containing 4.85 million pages.

A. Rationale

Previously developed data sets started with the idea of collecting pages around targeted classes so the resultant could be considered a class-driven data set. Even if the selection methodology was set to random, all pages out of the targeted classes were labeled with an OUT tag. Also some data sets were biased to popular pages or pages with rich content. We aimed to have a data-driven data set so we did not select a list of types beforehand, leaving the random selection of the pages to drive the judgment process. We have concerns with biasing previous data sets to head entities ignoring the long tail, which is the majority. We mean by tail entities, entities which have very limited features and are possibly marked as stubs in Wikipedia. So we decided to build our own data set with some constraints:

- Ensure randomness to represent the real case.
- Each page could have tags in different level of granularity. (e.g., Petros Christo is a guitarist, a musician, and a person).
- Each page can have multiple tags even in the same level of granularity. (e.g., Bob Golic is a footballer, an actor, and an announcer).

B. Data Set Building Process

We implemented a simple application that randomly selects a Wikipedia page and opens it in the browser to fasten the judgment process. The tool enables the judge to tag each page with multiple classes whether fine or coarse grained. We have managed to tag 2500 Wikipedia pages at a rate of 100 pages per hour after ramping up.

C. Entity Classes

We invistigated three ontologies: The extended named entity hierarchy [25], DBPedia ontology[1], and Schema.org[2]. Although Sekine's set considered a detailed one containing about 150 classes, lots of fine grained classes we faced in our judgment process were not covered. After we started using it, we experienced lots of misses, so we decide to switch to Schema.org. We have a very good experience with Schema.org as. By finishing our judgment phase over 2500 pages, we had 3 tiers of refinement and unification to make sure of consistent behaviors. The inter annotator agreement was around 96%. Our data set recorded 808 unique classes. Table I shows some selected types along with their densities (i.e., the number of pages labeled with that type).

D. Data Set Analysis

Our coarse-grained classes could be grouped as:

- Main classes like: persons, locations, organizations, events, products, awards, and species.
- Wikipedia specific classes like: disambiguation pages, list pages, and chronology pages.

[1] http://mappings.dbpedia.org/server/ontology/classes/ (January 2022)
[2] http://schema.org/docs/full.html (January 2022)

TABLE I. Sample classes and their densities from ClassifyWiki's data set (2500 Wikipedia articles).

Class [density]	Class [density]
award [8]	event > election [7]
event > crime - attack [6]	event > natural disaster [2]
event > sports competition > football [20]	event > war - battle [10]
location > architectural structure > building > hospital [1]	location > architectural structure > building > museum [8]
location > architectural structure > building > worship place > church [13]	location > architectural structure > infrastructure > line > road > highway [12]
location > architectural structure > infrastructure > station > railway station [21]	location > architectural structure > sports venue - stadium [5]
location > landform > mountain [15]	location > landform > river [19]
location > populated place > settlement > city [10]	location > populated place > settlement > village [135]
organization > company [43]	organization > company > bank [2]
organization > group > band - orchestra [34]	organization > local business > restaurant [4]
organization > educational institution > school [14]	organization > political organization > political party [9]
organization > ngo [3]	person > actor [47]
person > musician > guitarist [8]	person > politician [73]
person > sports player > athletics > track and field [17]	product > creative work > broadcast program > tv series [24]
product > creative work > book > novel [25]	product > creative work > music > song [35]
product > creative work > picture > painting [5]	product > vehicle > ship [13]
species > animal [106]	species > plant [30]

TABLE II. Number of unique classes for each targeted slice in ClassifyWiki data set, training set, development set, and test set.

Slice Num. Pages (>=)	All Set (2500)	Train Set (1000)	Dev. Set (1250)	Test Set (250)
50 pages	30	13	18	1
25 pages	46	25	27	5
10 pages	106	50	58	19
5 pages	201	103	112	36
1 page	808	543	535	216

- Generic class that represents a definition or a concept like: Solar zenith angle, Snow boating, and Focal length.

Since our target is to build a random set, we assessed the randomness of the data set by applying 2 heuristic rules:

- We noticed in the judgment phase that most of the pages of species class contained a taxobox template. By applying a simple heuristic to collect all pages with this template, we found 0.28M out of 4.85M (5.8%).
- As advised by the literature, person pages could be identified by persondata templates. Applying this rule resulted in 1.20M out of 4.85M (24.7%).

Table III summarizes the percentage of each one of our top 6 classes versus the whole set. We found a good correlation between the results of the 2 previous heuristics and the percentages table. We splitted our data set into 3 parts: 1000 pages for training, 1250 as a development set, and 250 as a blind one. As ClassifyWiki targets to build generic-type classifiers for any given input set while trying to minimize the training instances as much as possible, we sliced our data set putting a minimum threshold for the number of pages in any class in order to experiment separately with each slice. Table II summarizes the number of classes fulfilling the density range of each slice of interest.

TABLE III. Top ClassifyWiki data set classes' percentages.

Class	Percent	Class	Percent
Person	27%	Location	21%
Organization	9%	Event	13%
Product	6%	Species	6%

4. Feature Set

This section presents an exhaustive list of features collected from previous work in Wikipedia classification and also from some other sister tasks like fact extraction, linking, and disambiguation. The feature set could be classified into 2 main groups; local and global features. Also, we illustrate a study of the dominance of each feature across the whole dump.

A. Local Features

Local features are features extracted at the page level, from the page content.

1. Category Type: Introduced by Suchanek et al. [17]. Categories are tokenized and pos tagged then the first plural noun is selected and stemmed (e.g., "North African countries" Wikipedia category contains the page "Libya" so its type would be "country"). We noticed that some categories have a sort of coupling that we can leverage like "Buildings and structures", so we extended the rule to accept it.

2. Category Grams: n-gram representation of tokenized categories.

3. Header Grams: n-gram representation of tokenized sections and subsections.

4. Page Title Grams: n-gram representation of tokenized page titles after removing the disambiguation suffix (e.g., "1989 film" considered a disambiguation suffix for the page "Batman (1989 film)").

5. Disambiguation Suffix Grams: n-gram representation of tokenized disambiguation suffixes.

6. First Sentence Grams: n-gram representation of tokenized first sentences. Dakka and Cucerzan [20] observed that a human can judge a page using only the first sentence.

7. Page Text Grams: n-gram representation of tokenized page contents.

8. Infobox Grams: n-gram representation of tokenized infobox attributes.

9. Selected Templates: Whether the page contains each one of the 3 templates: coord (co-ordinates), persondata, and taxobox.

10. Sidebar Template Grams: n-gram representation of tokenized sidebars.

11. Definition Sentence Type: The head noun after the copula (is, are, was, or were) in the first sentence [19] (e.g., "Existere" has a definition sentence of "is a Canadian magazine" so its type would be "magazine").

12. Definition Sentence Grams: n-gram representation of the definition sentence, not to account only for the head noun but also to support classes like: "Nationality book", "Sport player", or "Genre song".

B. Global Features

These features are corpus level features extracted from the whole dump out of the page content.

1. Context Grams: n-gram representation of tokenized context. Context is represented by all sentences having a hyperlink to the page in hand.

TABLE IV. Feature dominance.

Feature	%	Feature	%
Category Type	84.6%	Header Grams	93.0%
Category Grams	93.9%	SideBar Grams	0.1%
Disamb. Suffix Grams	11.5%	Infobox Grams	50.9%
Page Title Grams	100.0%	Coord	14.4%
Page Text Grams	99.8%	PersonData	24.8%
First Sentence Grams	99.8%	Taxobox	5.8%
Def. Sentence Grams	77.4%	Context Grams	73.5%
Def. Sentence Type	77.4%	List Page Type	43.3%
List Page Grams	47.1%		

2. List Page Grams: n-gram representation for each tokenized list page title that includes the target article [23]. We remove "list, lists, and table of" markers before applying the rule (e.g., "List of Spanish-language authors" is a list page contains "Gutierre de Cetina").

3. List Page Type: Same heuristic adopted in YAGO was used to introduce the list page type (e.g., "Gutierre de Cetina" Wikipedia page could have the type "author" as it is contained in "List of Spanish-language authors").

C. Features Dominance

We examined the dominance of each feature across the whole dump to measure its effectiveness. Table IV presents each feature along with the percent of Wikipedia pages with this feature.

5. ClassifyWiki Approach

In this section, we present ClassifyWiki's modules in both training and testing modes. Also, we summarize our efforts in experimenting with each feature to assess its contribution. Finally, we introduce our incremental approach in combining the features till reaching out our final feature set.

A. ClassifyWiki Modules

The flow of our automated pipeline (see Figure 1) has 2 modes. A training mode in which we experimented to best tune our framework and a testing one which simulates the real time usage of the framework. The flow in the training mode is as follows. Wikipedia parser cleans the training set pages out of the XML dump. Feature extractor extracts the selected features for this experiment. Trainer accepts the positive set, generates a negative one, and launches the training process to build a new model. While in testing mode, the parser cleans the pages of the set. Feature extractor collects the same feature set as in the training of this experiment. The Evaluator accepts the featured set along with the model to label the set and generate an evaluation report.

1. **Wikipedia Parser**: Responsible for converting the raw XML dump into a clean form. It eliminates HTML tags, cleans Wikipedia specific markups, removes tables, cleans templates, and resolves hyperlinks. Useful structures removed by the parser are stored in a parallel stream

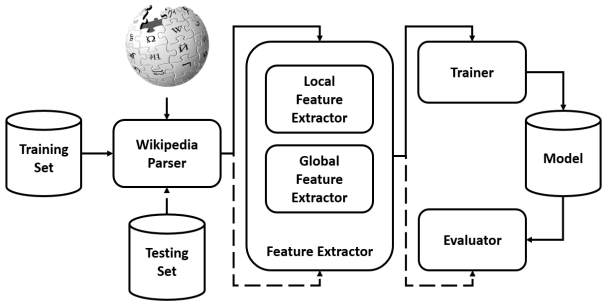

Figure 1. ClassifyWiki pipeline.

for other scenarios like templates. We experimented with several open source parsers, but we ended up implementing our own. Also, the parser runs through a pipeline of NLP components, sentence breaker, tokenizer, stemmer, and a POS tagger, in order to clean the text and prepare for the feature engineering step.

2. **Feature Extractor**: Implements the local and global features (Section 4). A normalization post processor was placed to normalize the case, if enabled, and to mask the numbers, replace digit sequences with a unified marker to reduce the sparsity.

3. **Trainer**: Owns the training process. It has 2 main steps: 1) Negative training data selection. 2) Launching the training process and serializing the model. As a baseline module for selecting negative training data, it selects random pages out of the whole dump in the same size of the positive training pages. Random selection showed experimentally its credibility.

4. **Evaluator**: Owns the evaluation process. Accepting a model and the featured test data, it launches the testing process and calculates the macro-averaged precision, recall, and f1-score for each slice of classes (Table II).

B. Feature Contribution

After building the baseline framework, our goal was to asses the value of each feature to avoid adding useless features. We conducted a series of experiments to build a classifier for each feature on its own then we compared them all in order that we can select a reduced feature set to reduce the dimensionality. Only Bhole et al. [12], to the best of our knowledge, reported using n-grams other than uni-grams. Effect of adding more n-grams to our feature set was addressed in ClassifyWiki. We checked the value of adding different combinations of n-grams (uni, bi, tri, uni + bi, bi + tri, uni + tri, and uni + bi + tri grams). A series of 7 experiments were conducted for each feature to detect the best n-grams to be used (Section 6-C).

C. Feature Selection

Forward feature selection was adopted to reduce the set of features in order to fit with a small number of training examples. We started to combine the features in one classifier from the best performing ones based on feature contribution's reports. In an incremental approach, we have added the features with the best n-grams representation one by one. We checked whether the newly added feature is adding value or just increasing the dimensionality without any gain (Section 6-D).

TABLE V. Different systems f1-scores on our test set.

System	>= 50	>= 25	>= 10	>= 5
Tkachenko et al. [23]	73%	61%	55%	36%
Saleh et al. [22]	76%	64%	50%	33%
Tardif et al. [24]	60%	52%	35%	21%
Nothman et al. [7]	70%	58%	48%	33%
Dakka & Cucerzan [20] (Baseline)	30%	25%	16%	11%
Dakka & Cucerzan [20] (Full Set)	38%	31%	19%	11%
Bhole et al. [12]	31%	26%	17%	10%
ClassifyWiki (Baseline)	53%	41%	29%	19%
ClassifyWiki (Final)	83%	67%	58%	40%

6. Experiments and Results

In this section, we introduce our baseline results, our experience with simulating lots of previous systems, features contribution, and our incremental approach in combining the features. Finally, we report the results of testing our final ClassifyWiki's implementation.

A. Previous Techniques

Although no previous system targeted a generic platform to generate binary classifiers given tens of positive training pages, we tried to simulate the behavior of each previous system's feature set. Most of the systems approached Wikipedia classification as a text classification task adopting an SVM as the classification algorithm. We learned from almost all of them that SVM is empirically the best one to be used. We had many variables: different test/training sets and different tagging guidelines which might lead to unfair comparisons so we decided to unify the comparison platform as follows:

- Train and test all systems using our dataset.
- Use the same SVM classification algorithm.
- Use the same module for negative data selection per class.
- The only variable would be the feature set.

It worth mentioning here that as all previous systems started with a set of classes in hand, there was no problem in how to collect negative training data. Simply the training data for other classes acted as negative for one class. ClassifyWiki introduces the idea that the consumer should only provide tens of positive training pages. Using our

TABLE VI. Feature contribution f1-scores on our dev. set - Best 3 bolded.

Feature Group	Best Grams	>= 50	>= 25	>= 10	>= 5
Category Type	Uni	52%	44%	37%	29%
Category	Uni/Bi	**60%**	**52%**	**43%**	**33%**
Headers	Uni	37%	31%	25%	16%
First Sent.	Uni/Bi	**57%**	**50%**	39%	27%
ListPage Type	Uni	13%	10%	8%	7%
List Page	Uni/Bi	12%	10%	8%	7%
Disamb. Suffix	Uni	7%	5%	6%	4%
Page Title	Uni	13%	14%	11%	9%
Def. Sent.	Uni	**56%**	**50%**	**44%**	**31%**
Def. Sent. Type	Uni	55%	**50%**	**43%**	**31%**
Infobox	Uni	**56%**	43%	41%	**32%**
Templates	Uni	32%	21%	14%	8%
Context	Uni	10%	8%	7%	5%
Page Text	Uni/Bi	30%	22%	13%	9%

automated platform, we had an experiment with each of the feature sets proposed by previous systems. All features were added with different prefixes as each system advised after tokenization, stemming, and normalization (lower casing and numbers masking) as a bonus step from ClassifyWiki. As we are interested in minimizing the number of training instances needed, we reported the macro-averaged f1-score of each system across different slices based on the number of positive training instances (Table II). Table V summarizes the scores of each slice using our test set (250 pages).

B. *ClassifyWiki Baseline*

To introduce our baseline, we just collected all the features of the previous systems to be our feature set. Table V illustrates our baseline results with the tag "ClassifyWiki (Baseline)" on the test set (250 pages).

C. *Feature Contribution*

Table VI illustrates our efforts analyzing the contribution of each feature. The table presents the features one by one along with the best n-gram representation with macro-averaged f1-scores for each slice based on the positive training pages size on our development set. Simply, we can find that the best performing features are the ones that represent fine grained types like categories, infoboxes, and definition sentences. Worth mentioning that category n-grams performed better than category type. For instance, railway station, power station, and radio station, 3 classes in our dataset, all had the same type, station, when category type feature was used.

D. *Feature Selection*

Table VII summarizes our set of experiments to combine all features in an incremental approach starting from the best performing ones based on Table VI. We report macro-averaged f1-scores for each slice based on the positive training data size over our development set (1250 pages).

TABLE VII. Features combinations f1-scores on our dev. set - Best bolded.

#	Features	>= 50	>= 25	>= 10	>= 5
1	Category + Infobox	65%	55%	46%	36%
2	1 + First Sent.	74%	59%	50%	37%
3	2 + Category Type	75%	62%	52%	38%
4	3 + Def. Sent.	76%	63%	53%	40%
5	4 + Def. Sent. Type	76%	65%	54%	41%
6	5 + Headers	77%	65%	55%	42%
7	6 + Templates	**78%**	**66%**	**55%**	42%
8	7 + Page Text	52%	40%	26%	17%
9	7 + Context	59%	50%	38%	25%
10	7 + Page Title	76%	65%	**55%**	**43%**
11	7 + List Page	**78%**	65%	54%	41%
12	7 + List Page Type	**78%**	**66%**	**55%**	42%
13	7 + Disamb. Suffix	**78%**	**66%**	**55%**	**43%**

TABLE VIII. Final results on our test set.

Metric	>= 50	>= 25	>= 10	>= 5
Precision	96%	83%	71%	48%
Recall	74%	59%	55%	39%
F1-Score	83%	67%	58%	40%

The finally selected feature set for ClassifyWiki noted with index 13. Degradation in numbers could be easily tracked for features with lots of noisy terms like page text and context features specially with small training sets.

The last step, the 13th one, grouped almost all of the classes around the average band. We set the objective of the system to have good results for any class not to tune it for head classes, that is why we chose macro-average scheme. Almost all of the classes had deep regressions in steps eight and nine related to page text and contextual features which added lots of noisy features. We need to invest more efforts to leverage those features by picking just the important terms out of them. We see that this is acceptable when we target fine grained classes especially with low sized input training set. Effective gains showed up early in the curve in the second or the third steps while minimal additions can be noticed at the right hand side. The cause behind this observation is that, we started integrating the best performing features at first as Table VI suggested. Footballers (person > sports player > football), disambiguation, and settlements (location > populated place > settlement) could be considered showcases in the way that the feature design boosted their f1-scores. Other ones like actors (person > actor) sacrificed for the sake of the whole framework. In the final step, the actors class is not at its best score while ClassifyWiki average score was getting better so we had to sacrifice some classes in order to favor the whole system.

E. Final ClassifyWiki System

We report our final scores on the test set (250 pages) in Table V using the tag "ClassifyWiki (Final)" and with more details in Table VIII. Tables (V, VIII) report on the test set while Tables (VI, VII) on the development set. Although the test set is pretty smaller than the development one, the results are in the same range.

Table IX summarizes final f1-scores for sample classes on our development set. The level of granularity affects the classification accuracy. The table shows that evaluation scores of some very specific classes like schools, albums, and movies were extremely high while they were low for generic or complex classes like sports competitions, companies, and singers. In the same logical group (i.e. professions), ClassifyWiki was able to build a reliable classifier for baseball players, a moderate one for politicians, and a weak one for singers.

TABLE IX. F1-Scores for sample entity classes over our test set (250 articles).

Entity Class	F1-Score
event > sports competition > football	1.00
location > architectural structure > infrastructure > road	1.00
organization > educational institution > school	1.00
person > sports player > athletics	1.00
person > sports player > baseball	1.00
product > creative work > broadcast program > tv series	1.00
product > creative work > game	1.00
product > creative work > movie	1.00
product > creative work > music > album	1.00
species	1.00
person	0.97
species > animal	0.97
person > sports player	0.94
location > populated place	0.92
location > populated place > settlement > village	0.91
person > sports player > football	0.89
disambiguation	0.88
species > animal > insect	0.87
location > populated place > settlement	0.85
product > creative work > music	0.85
product	0.77
product > creative work	0.68
location > architectural structure > infrastructure > station	0.67
organization > educational institution	0.67
organization > group > band - troupe - chorus - orchestra	0.67
organization > sports club > football	0.67
person > lawyer	0.67
organization	0.65
event	0.63
person > actor	0.57
product > creative work > book	0.55
product > creative work > music > song	0.55
location > architectural structure > building > worship place	0.50
person > politician	0.50
product > creative work > book > novel	0.50
person > musician	0.43
species > plant	0.40
event > sports competition	0.33
organization > company	0.29
person > musician > singer	0.18

7. Conclusion

We implemented a framework to build classifiers given any set of pages in any level of granularity possibly in any size. The more the size of the input set, the better the resulted classifier's accuracy. We report 83% macro-averaged f1-score using 50 positive training instances. We tested our framework over more than 100 entity classes using our dataset based on schema.org. Although schema.org was used to evaluate the framework over more than 100 classes, ClassifyWiki is not bounded by it nor Wikipedia categorization system, as a schema matching task. In real time, the input training data is left to the user whether to follow the ontology or not as our output is not models for some classes but a framework to build models. ClassifyWiki does not learn some specific classes like all previous systems but, theoretically, it can generate classifiers for any entity class.

8. Future Work

Bootstrapping techniques could be experimented to boost our recall [7]. Some thoughts could be subject for experimentation: 1) Number of bootstrapping rounds. 2) Whether to use all features in the same round or one feature per round 3) Selection of new data points for the next round to be from positive, negative, border line, or confident ones.

ClassifyWiki's method to select the negative training instances, random sampling, could be enhanced. Although it models the whole spectrum, it lacks modeling border line cases. For instance, to model football players, the representation of sister entity classes like basketball players will not be that big. Adding more instances from sister types in the same generic one as negative data could help.

For feature engineering, we can leverage hearst patterns, definites, apposition and copula, common pronoun to enable the gender discrimination, and the most frequent "the XX" pattern [26], [4]. Hypernym discovery methods could be helpful also [27], [28]. We did not experiment with tables [29]. We need to invest more in contextual

features, especially by adding more hyperlinks [4] or using a disambiguator [30]. By adding more hyperlinks, the context could grow more and more enabling a better representation.

Some sort of comparisons could be tackled to test ClassifyWiki against some knowledge bases like DBPedia to assess its edge. Potential gains could be summarized in better recall and classes which are not supported in the knowledge base, user-defined classes.

As we experience ClassifyWiki's potential, we would like to formaly evaluate the whole experience in many Wikipedia languages using language links [5], [21]. Collecting a very small list of museums in English, could result in a French or Chinese museum extractor.

Another potential direction is to plug ClassifyWiki in different NLP tasks. We plan to automate the process of building multilingual fine-grained NERs (museum, road, church, footballer, guitarist) and user-defined ones (worship place, sports-person, musician). The missing block is a framework to generate many pages around this class. Then, we can tag hyperlinks to produce NER training data.

In parallel, we plan to experiment with generic text classification, topic modeling, in any language. With samples of few entity classes in the topic of interest, ClassifyWiki can get thousands of pages to be mapped to many languages through the language links acting as training data for topic modeling, if sufficient, or after using them as seeds for another round in the target language.

References

[1] J. Hoffart, D. Milchevski, and G. Weikum, "Aesthetics: analytics with strings, things, and cats," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, 2014, pp. 2018–2020.

[2] X. Ling and D. S. Weld, "Fine-grained entity recognition," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[3] R. Sutoyo, C. Quix, and F. Kastrati, "Factrunner: A new system for nlp-based information extraction from wikipedia," in *International Conference on Web Information Systems and Technologies*. Springer, 2013, pp. 225–240.

[4] F. Wu and D. S. Weld, "Open information extraction using wikipedia," in *Proceedings of the 48th annual meeting of the association for computational linguistics*, 2010, pp. 118–127.

[5] F. Mahdisoltani, J. Biega, and F. Suchanek, "Yago3: A knowledge base from multilingual wikipedias," in *7th biennial conference on innovative data systems research*. CIDR Conference, 2014.

[6] V. Nastase and M. Strube, "Transforming wikipedia into a large scale multilingual concept network," *Artificial Intelligence*, vol. 194, pp. 62–85, 2013.

[7] J. Nothman, J. R. Curran, and T. Murphy, "Transforming wikipedia into named entity training data," in *Proceedings of the Australasian Language Technology Association Workshop 2008*, 2008, pp. 124–132.

[8] J.-B. Michel, Y. K. Shen, A. P. Aiden, A. Veres, M. K. Gray, G. B. Team, J. P. Pickett, D. Hoiberg, D. Clancy, P. Norvig *et al.*, "Quantitative analysis of culture using millions of digitized books," *science*, vol. 331, no. 6014, pp. 176–182, 2011.

[9] M. N. Gerguis, C. Salama, and M. W. El-Kharashi, "Wikitrends: Unstructured wikipedia-based text analytics framework," in *International Conference on Applications of Natural Language to Information Systems*. Springer, 2017, pp. 45–57.

[10] P. Wang, J. Hu, H.-J. Zeng, and Z. Chen, "Using wikipedia knowledge to improve text classification," *Knowledge and Information Systems*, vol. 19, no. 3, pp. 265–281, 2009.

[11] M. N. Gerguis, C. Salama, and M. W. El-Kharashi, "Asu: An experimental study on applying deep learning in twitter named entity recognition." in *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, 2016, pp. 188–196.

[12] A. Bhole, B. Fortuna, M. Grobelnik, and D. Mladenic, "Extracting named entities and relating them over time based on wikipedia," *Informatica*, vol. 31, no. 4, 2007.

[13] C. Bøhn and K. Nørvåg, "Extracting named entities and synonyms from wikipedia," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*. IEEE, 2010, pp. 1300–1307.

[14] A. E. Richman and P. Schone, "Mining wiki resources for multilingual named entity recognition," in *Proceedings of ACL-08: HLT*, 2008, pp. 1–9.

[15] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann, "Dbpedia-a crystallization point for the web of data," *Journal of web semantics*, vol. 7, no. 3, pp. 154–165, 2009.

[16] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 1247–1250.

[17] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 697–706.

[18] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: A large ontology from wikipedia and wordnet," *Journal of Web Semantics*, vol. 6, no. 3, pp. 203–217, 2008.

[19] K. Torisawa *et al.*, "Exploiting wikipedia as external knowledge for named entity recognition," in *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007, pp. 698–707.

[20] W. Dakka and S. Cucerzan, "Augmenting wikipedia with named entity tags," in *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*, 2008.

[21] J. Zhou, B. Li, and Y. Tang, "Classifying articles in chinese wikipedia with fine-grained named entity types," *Journal of Computing Science and Engineering*, vol. 8, no. 3, pp. 137–148, 2014.

[22] I. Saleh, K. Darwish, and A. Fahmy, "Classifying wikipedia articles into ne's using svm's with threshold adjustment," in *Proceedings of the 2010 Named Entities Workshop*, 2010, pp. 85–92.

[23] M. Tkachenko, A. Ulanov, and A. Simanovsky, "Fine grained classification of named entities in wikipedia," *HP Laboratories Technical Report-HPL-2010-166*, 2010.

[24] S. Tardif, J. R. Curran, and T. Murphy, "Improved text categorisation for wikipedia named entities," in *Proceedings of the Australasian Language Technology Association Workshop 2009*, 2009, pp. 104–108.

[25] S. Sekine, K. Sudo, and C. Nobata, "Extended named entity hierarchy." in *LREC*, 2002.

[26] P. Cimiano, S. Handschuh, and S. Staab, "Towards the self-annotating web," in *Proceedings of the 13th international conference on World Wide Web*, 2004, pp. 462–471.

[27] S. P. Ponzetto and M. Strube, "Taxonomy induction based on a collaboratively built knowledge repository," *Artificial Intelligence*, vol. 175, no. 9-10, pp. 1737–1756, 2011.

[28] A. Ritter, S. Soderland, and O. Etzioni, "What is this, anyway: Automatic hypernym discovery." in *AAAI Spring Symposium: Learning by Reading and Learning to Read*, 2009, pp. 88–93.

[29] C. S. Bhagavatula, T. Noraset, and D. Downey, "Methods for exploring and mining tables on wikipedia," in *Proceedings of the ACM SIGKDD workshop on interactive data exploration and analytics*, 2013, pp. 18–26.

[30] R. Mihalcea and A. Csomai, "Wikify! linking documents to encyclopedic knowledge," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, 2007, pp. 233–242.

Michel Naim Gerguis Michel received his B.Sc. and M.Sc. in computer engineering from Ain Shams University (ASU), Cairo, Egypt in 2012 and 2017 respectively. He is a data and applied science manager in Microsoft Egypt. His research internets and experiences span natural language processing, data analytics, social media, and knowledge extraction. He is a fan of building new production systems that deliver insights based on textual big data. Currently, Michel leads Microsoft Clarity data science team.

M. Watheq El-Kharashi M. Watheq El-Kharashi received the Ph.D. degree in computer engineering from the University of Victoria, Victoria, BC, Canada, in 2002, and the B.Sc. degree (first class honors) and the M.Sc. degree in computer engineering from Ain Shams University, Cairo, Egypt, in 1992 and 1996, respectively. He is a Professor in the Department of Computer and Systems Engineering, Ain Shams University, Cairo, Egypt and an Adjunct Professor in the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada. His general research interests are in advanced system architectures, especially Networks-on-Chip (NoC), Systems-on-Chip (SoC), and secure hardware. He published about 100 papers in refereed international journals and conferences and authored two books and 6 book chapters.

Cherif Salama Cherif Salama received his B.Sc. and M.Sc. degrees from the computer and systems engineering department of Ain Shams University (ASU) in 2001 and 2006 respectively. In 2010, he received his Ph.D. degree in computer science from Rice University, Houston, Texas. He worked as assistant professor in the Computer and Systems Engineering Department of ASU and as adjunct lecturer in the EELU, the MIU, and the GUC. He served as unit head of the Computer Engineering and Software Systems program at Ain Shams University (ASU). He is currently an associate professor in the Computer Science and Engineering Department at The American University in Cairo. He was also the key person in the design and development of the Verilog Preprocessor funded by Intel through the SRC. His research interests and publications span a wide spectrum including computer architecture, CAD, hardware description languages, programming languages, parallel computing, and AI.