Pythia: A System for Online Topic Discovery of Social Media Posts

Iouliana Litou
Department of Informatics
Athens University of Economics and Business
Email: litou@aueb.gr

Vana Kalogeraki
Department of Informatics
Athens University of Economics and Business
Email: vana@aueb.gr

***Abstract*—Social media constitute nowadays one of the most common communication mediums. Millions of users exploit them daily to share information with their community in the network via messages, referred as *posts*. The massive volume of information shared is extremely diverse and covers a vast spectrum of topics and interests. Automatically identifying the topics of the posts is of particular interest as this can assist in a variety of applications, such as event detection, trends discovery, expert finding etc. However, designing an automated system that requires no human agent participation to identify the topics covered in posts published in Online Social Networks (OSNs) presents manifold challenges. First, posts are unstructured and commonly short, limited to just a few characters. This prevents existing classification schemes to be directly applied in such cases, due to sparseness of the text. Second, new information emerges constantly, hence building a learning corpus from past posts may fail to capture the ever evolving information emerging in OSNs. To overcome the aforementioned limitations we have designed *Pythia*, an automated system for short text classification that exploits the Wikipedia structure and articles to identify the topics of the posts. The topic discovery is performed in two phases. In the first step, the system exploits Wikipedia categories and articles of the corresponding categories to build the training corpus for the suppervised learning. In the second step, the text of a given post is augmented using a text enrichment mechanism that extends the post with relevant Wikipedia articles. After the initial steps are performed, we deploy k-NN classifier to determine the topic(s) covered in the original post.**

I. Introduction

Social media platforms have remarkably grown over the past years. Millions of people are using them daily to communicate with their network of friends using messages referred as *posts*. Different social platforms have emerged, offering diverse communication types, i.e., structure of posts. For instance, Instagram is used to share photos/videos, Facebook and Twitter to share information containing variable features such as text, photos, videos, etc. Over 2.3 billion users worldwide are subscribed to a social platform service[1] and produce a massive volume of content via publishing posts, e.g., consider the phenomena of Facebook with users sending over 31.25 million messages and viewing 2.77 million videos hourly[2].

The information shared in OSNs covers a vast spectrum of topics. Identifying the topics a post refers to can be useful in a number of applications and various domains. For instance, it can be utilized in applications for trend discovery, expert identification or detection of influential users based on the topics of interest , to name a few. Although the topic identification is immensely useful and interesting, it is also extremely challenging. Aside from the diverse structure of the posts in Online Social Networks (OSNs), there is also usually a common feature present to all services that makes the task of topic discovery remarkably hard, i.e., the short text that accompanies the post. Users rarely spent time writing long stories, as the type of information they share is usually instant or spontaneous. Moreover, when the posts contain some short of media, the text usually serves as a comprehensive description of the content. Therefore, the sparseness of the text prohibits the deployment of traditional text-classification approaches to be directly applied in the case of OSN posts. Apart from the concise text, posts contain various non-standardizations and misspellings that further hinder the application of conventional machine learning and text mining algorithms.

In this demo we present *Pythia*, a system that addresses the aforementioned limitations. The purpose of the demo is to show how we can efficiently build an autonomous classification schema for posts in OSNs with the use of Wikipedia and traditional text-classification algorithms and later use it to infer the attributes that make certain post more popular, such as the use of certain words or mentioning specific users. Wikipedia [1] is a free online encyclopedia that is written collaboratively. We use Wikipedia in two ways, (i) to build the training corpus of the application and (ii) to handle the sparseness presented in the posts in OSNs. Wikipedia contains articles organized in various taxonomies, called categories. Therefore, Wikipedia provides an ideal training corpus regarding text classification, as it consists of documents, i.e., articles, that are already classified and therefore it has been exploited as such in prior works [2], [3]. Apart from extracting the set of classified documents, we additionally use Wikipedia to crawl articles relative to the posts and augment the text of the post with the corresponding articles. Therefore, we construct and *enrich* the post. We crawl articles based on both the *Named Entities* that may be present on the posts and the *lemmas*. Furthermore, during the queries for the related articles, we allow for redirects. That assists in addressing two important issues, (i) capturing and correcting common misspellings, e.g., while querying for *"laywer"*, we retrieve the article referring to *"lawyer"* and (ii) entity abbreviations that are commonly used in social media are replaced with the full name and the corresponding article is retrieved, e.g., a query for the term *"PhD"* will result in the article *"Doctor of Philosophy"*.

[1]https://www.statista.com/topics/1164/social-networks/

[2]http://www.cio.com/article/2915592/social-media/7-staggering-social-media-use-by-the-minute-stats.html#slide2

After the post is enriched with the corresponding articles, a traditional text-classification method is deployed to identify the topics of the post along with a *degree of confidence*. We note that posts may cover a number of topics, therefore we identify all the topics it may refer to along with the confidence of the prediction. In our work we use k-Nearest Neighbors classifier with cosine similarity to identify the articles that are closer to the enriched post and extract the relevant topics. This demo is built upon our earlier work [4] where we have have implemented a lazy classification approach to identify topics of tweets and infer the popularity features based on the topics. Similarly to [4] we use Wikipedia to enrich tweets, but we extend this work to consider Named Entities and the application of traditional text-classification methods to identify the topics of the posts.

Authors in [2] address the problem of short-text classification by enriching texts through topics of multi-granularity and use Wikipedia for the training corpus. However their approach is too elaborate and cannot be applied for real-time, online topic discovery. Moreover, it requires the tuning of certain parameters that affect the classification results and no schema is proposed to automatically identify the most appropriate values. Finally, non-standardization and misspellings are not considered in the input text. A similar approach is also used in [3]. In [5] the authors cluster posts of the network of Twitter, i.e., *tweets*, into predefined topics, yet the proposed approach requires a prior set of tweets with identified topics for the task, a restriction we overcome

The contributions of this work are summarized as follows:

- We propose an enrichment methodology for posts in OSNs that overcomes two important restrictions, i.e., the common misspellings and use of abbreviations, while addressing the sparseness of a post and making it more descriptive by augmenting it with related Wikipedia articles. Our proposed approach also considers Named Entities presented in the posts.
- The proposed approach provides a radical advantage by allowing the use of common text-classification models to be applied in the case of the post in the social media. Additionally, it requires no training corpus to be built a priory, as it exploits an existing one, i.e., the Wikipedia category taxonomy. We note that this also provides the ability to define a number of topics and refined levels of classification.
- We build a classification schema that not only identifies the topics of the posts, but further estimates a confidence score. Aside from identifying relevant topics we also provide a probability that expresses the likelihood of a post belonging to a certain topic. Our preliminary experimental evaluation proves that the proposed methodology is able to correctly identify topics of several posts among different categories.

The rest of the paper is organised as follows. In Section II we present in detail the Pythia for online topic discovery of posts in social media using Wikipedia and provide screenshots and results of the Pythia system that showcase the performance of the proposed approach in Section III.

II. System Description

In this section we provide a detailed description of *Pythia*, our proposed classification schema for the OSN posts.The overall system comprises three components, (i) a component that builds the training corpus, (ii) a post enrichment component and (iii) the classification component. First we introduce Wikipedia and the API we exploit to crawl the articles. We then describe the construction of the training corpus using Wikipedia and later present the enrichment methodology. Finally we discuss the classification approach to identify the topics covered in the posts.

A. Wikipedia Crawler

Wikipedia is a free online Encyclopedia. Articles in Wikipedia are written collaboratively and are constantly improved and updated[3]. Articles are organized under a taxonomy of different topics, called categories and each article may belong to one or more categories [4]. Wikipedia offers an API that allows to query various information regarding articles and categories in different formats, e.g., XML or JSON [6]. We use this API to crawl the articles of the different categories and articles based on the Named Entities and lemmas of the posts. An example query to get the articles of a category, e.g., *Arts*, using the API, is the following

```
https://en.wikipedia.org/w/api.php?action=query&list=
categorymembers&cmlimit=100s&redirects&cmtype=page&
format=json&cmtitle=Category:Arts
```

The JSON returned for the above query is of the format

```
{"batchcomplete":"","query":{"categorymembers":[
{"pageid":752,"ns":0,"title":"Art"},
{"pageid":29560452,"ns":0,"title":"The arts"},
{"pageid":2583747,"ns":0,"title":"Work of art"},
{"pageid":1480241,"ns":100,"title":"Portal:Arts"},...]}}
```

Similarly, an example to query the articles based on their title, e.g., the articla entitle *Art*, is the following

```
https://en.wikipedia.org/w/api.php?action=query&format=json
&prop=extracts&redirects&titles=Art
```

From the JSON returned by such a query we extract the main article, if there exists one, and strip the HTML tags presented in the body of the article. Note that we allow for redirects by setting the property *redirects*, in order to correct any misspellings or get full articles for common abbreviations.

B. Training corpus

Given a set of categories C, the Wikipedia articles under these categories are collected and labeled according to the corresponding category. In this work we considered the categories presented in Table I. We consider these categories to cover a vast range of the topics discussed in Social Networks, we note however that these are indicative and independent of the classification process, therefore various or even more refined categories may be defined, depending on the application. As some categories may have hundreds of articles, while others significantly fewer, we limit the articles per category to a maximum of 100 articles, using the *cmlimit* parameter. Once

[3]https://en.wikipedia.org/wiki/Wikipedia:Introduction
[4]https://en.wikipedia.org/wiki/Wikipedia:FAQ/Categorization

Arts	Belief	Business	Computing
Culture	Disasters	Education	Electronics
Entertainment	Engineering	Events	Foods
Games	Geography	Health	History
Humanities	Industry	Law	Library science
Mass media	Nature	People	Personal life
Philosophy	Politics	Self	Science
Society	Sports	Statistics	Technology

TABLE I. TOPICS OF THE TRAINING CORPUS

the corresponding articles are crawled, a text pre-processing is performed and the document vectors are constructed. That is, for each article we exclude stopwords and using the Stanford CoreNLP API [7], we produce the lemmas of the text. Stanford CoreNLP provides a set of natural language analysis tools. Finally, Term-Frequency and Inverse Document Frequency (TF-IDF) values are estimated [5].

C. Post Enrichment

Given as input the post whose topics we aim to identify, the post enrichment component is triggered. A network connection is required to query Wikipedia, as the enrichment component is executed online. Given the row text, we initially perform two separate task, (i) Named Entity recognition and (ii) Lemmatization and stopwords removal. In order to extract the Named Entities that are present in the original text, we use the Stanford Named Entity Recognizer [8] that follows the model proposed in [9]. For the Lemmatization we use the Stanford CoreNLP API, as previously. After the Named entities and the lemmas have been produced, we query Wikipedia for the corresponding articles based both on the Named Entities and the lemmas. If an article is found, then we crawl it. When all articles relevant to the post are gathered, a single document is produced as the union of all the articles. Given the aggregated document we again exclude stopwords and perform lemmatization to construct the final document vector to be used in the classification process. We refer to the final document vector as *Enriched Post*. The overall procedure is depicted in Figure 1.

D. Post Classification

Having built the Enriched Post, the last step is to identify the topics covered by the post. For the classification process we use the k-NN classifier and a confidence metric that determines the probability of a post belonging on a certain category, based on the nearest neighbors identified by the k-NN classifier.

1) kNN-Classifier: Given the TF-IDF values of the articles in the training corpus and their categories, we use cosine similarity to estimate the distance of the Enriched Post to the articles in the corpus. That is, given the Enriched Post E, for each term $e_i \in E$ we multiply the normalized term frequency value of e_i in E with its IDF value to get the TF-IDF score and finally compute the cosine similarity of E to a document in the corpus D as

$$similarity(E,D) = \frac{\sum_{i=1}^{n} E_i D_i}{\sqrt{\sum_{i=1}^{n} E_i}\sqrt{\sum_{i=1}^{n} D_i}}$$

[5]https://en.wikipedia.org/wiki/Tf-idf

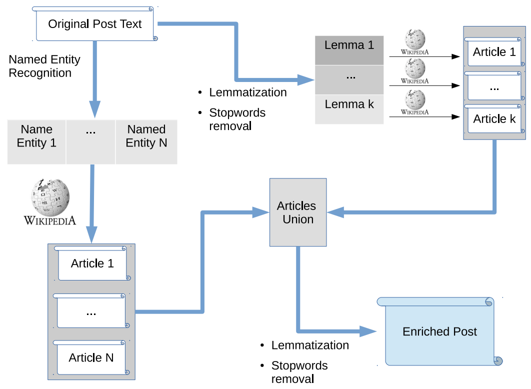


Fig. 1. Overview of the Post Enrichment Component

where n is the total number of terms in both E and D and E_i and D_i are the TF-IDF values of the term i in E and D respectively. Once the similarities are computed, the kNN-classifier identifies the k nearest neighbors and determines the categories of the original post based on the categories of the k neighbors. In our experiments we have set k equal to 20, however this can be set otherwise. After the nearest neighbors are identified, we propose the use of a metric that estimates the probability of a post belonging to a topic and identify the confidence of the classification.

2) Confidence score: Given the k nearest neighbors and their cosine similarity, we aim at determining the probability of the post being relevant to the topic. To achieve this, we initially sum the similarity scores of the k nearest articles belonging to the same category. That is, for each category c among the k neighbors, we estimate the function $f(c)$ as

$$f(c) = \sum_{D_i \in c} similarity(E, D_i)$$

where D_i is a document in the near neighborhood belonging in category c. Given the $f(c)$, if the maximum value is greater than 1, i.e., $max\{f(c)\} > 1, \forall c \in C$, then to get the final confidence score $p(E,c)$, we divide all $f(c)$ values with the maximum value, otherwise we set $p(E,c) = f(c)$, that is

$$p(E,c) = \begin{cases} if(max\{f(c)\} > 1) : f(c)/max\{f(c)\} \\ else : f(c) \end{cases}$$

Finally, we exclude categories that have low confidence scores. In our experiments we set a confidence threshold of $\theta = 0.45$, and we only keep categories that have a $p(E,c) > \theta$.

E. Popular terms

We have further implemented a component that identifies popular terms related to the post. The component exploits the Twitter Streaming API [6] to stream tweets containing at least one of the terms presented in the post. We set a popularity threshold ϱ and based on the retrieved tweets whose retweets exceed ϱ, we extract the most frequent terms. The terms are presented in a word cloud (see Section III for more details).

III. DEMO DESCRIPTION

Equipment. We will be using a laptop to demonstrate Pythia. The time to build the training corpus is approximately

[6]https://dev.twitter.com/streaming/overview

Example 1	As Trump Orders Wall, Mexico's President Considers Canceling U.S. Trip, via @nytimes https://t.co/m6JiaB2bAI	Politics: 1.0, Humanities: 0.8, Mass media: 0.46
Example 2	RT @nytimes: That old, unsecured Android phone Donald Trump uses for Twitter could be an opening to security threats	Mass media: 1.0
Example 3	I've collected 31,600 gold coins! https://t.co/aZ16e0PHvg #android, #androidgames, #gameinsight	Industry: 0.84
Example 4	RT @_Creative_World: Please don't #text & #drive! :-) #CreativeWorld #useful #grok #Zen #socialmedia - #thc #ufo #alternative 2 #ai ...	Engineering: 1.0, Belief: 0.66
Example 5	Yamaha YPT240 61-Key Portable Keyboard with Ultra Wide Stereo, iPhone, iPad and iPod touch.. https://t.co/ZPkRnUSVcc	Technology: 1.0, Mass media: 0.68, Industry: 0.46
Example 6	Press: [Tech Times] 'Carpool Karaoke': James Corden's Top Rides Adele, Bieber, Selena, Sia, One... https://t.co/0HQvdDtEbT	Mass media: 0.87, Personal life: 0.66, Culture: 0.65, Humanities: 0.46, Arts: 0.46
Example 7	haha tell the nurse to take away your iPhone and wheel you back to the duck pond, grandpa https://t.co/4l6TrrfBKM	Health: 0.6, Personal life: 0.5, Humanities: 0.5
Example 8	I've harvested 516 food! Check your patches for food too! https://t.co/zca6DARMKr #android, #androidgames, #gameinsight	Foods: 1.0
Example 9	Top story: Facebook brings its Slideshow movie-maker to Android | TechCrunch https://t.co/Sg2IHDCJfR, see more https://t.co/2OWUYgbEkC	Mass media: 1.0, Technology: 0.74, Library science: 0.45
Example 10	Our voices reveals lots of things, illnesses included #vtecl #machinelearning #health https://t.co/tjPbH2oiZh	Health: 1.0, Personal life: 0.64, Culture: 0.64, Belief: 0.63, Humanities: 0.46

TABLE II. CLASSIFICATION EXAMPLES

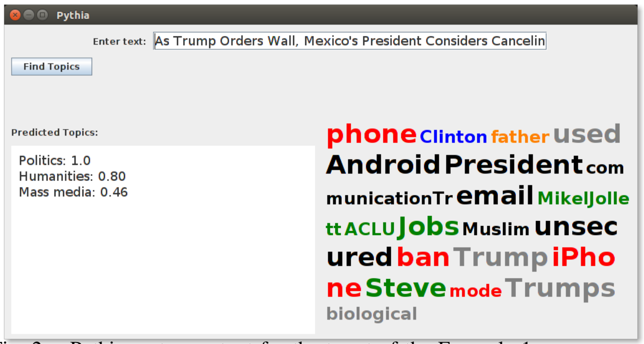
3 minutes, depending on the categories and the number of documents. The online classification requires less than 10 seconds to query, crawl and process the Wikipedia articles. A network connection is required to query Wikipedia for relevant articles in the enrichment component as the query to Wikipedia is performed online in real-time.

Demo Plan. Users will be able to interact with Pythia through the application interface. Pythia is implemented in Scala and we have evaluated it in posts of Twitter. In Table II we present different examples of the input that the users will be able to use in our system and their corresponding classification results after the execution. Overall, it can be seen that the classification works pretty accurately and does so completely automatically, with just the use of Wikipedia and without any assistance, such as keywords denoting certain categories. Probably the most outstanding example illustrating the efficiency of the Pythia classification process is the case of the Example 4. Pythia manages to identify that the tweet refers to Engineering, i.e., *alternative to AI*, while conveying a common Belief, i.e, *don't text and drive*.

In Figures 2 and 3 we present a snapshot of our Pythia system for the Examples 1 and 10 respectively. In addition to the topics predicted, the users can also see a word cloud. The word cloud depicts the most frequent words presented in popular tweets related to the input post text. We extract these words based on a dataset of approximately 2M tweets collected via Twitter Streamer API on January of 2017. We set the popularity threshold ϱ to 200. Tweets with over 200 retweets that contain words presented in the input tweet text are retrieved and the 20 most frequently words appeared among them are used to construct the word cloud. Therefore, the users are able to see what was trending regarding the topic presented in the tweet. Notice that we also get names of tweet accounts, e.g., *MikelJollett*, denoting that a tweet of the specific account was frequently retweeted and the *RT* $@Mikel\_Jollett$ appeared in many related tweets.

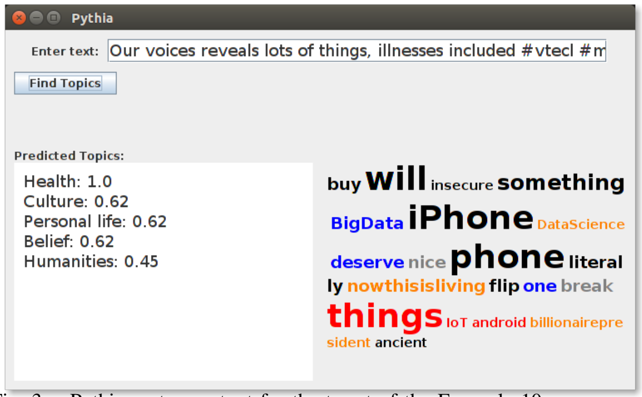
ACKNOWLEDGEMENT

This research has been financed by the European Union through the FP7 ERC IDEAS 308019 NGHCS project and the Horizon 2020 688380 VaVeL project.


Pythia
Enter text:
As Trump Orders Wall, Mexico's President Considers Cancelin
Find Topics
Predicted Topics:
phone Clinton father used Android President com municationTr email MikelJolle tt ACLU Jobs Muslim unsec ured ban Trump iPho ne Steve mode Trumps biological

Politics: 1.0
Humanities: 0.80
Mass media: 0.46

Fig. 2. Pythia system output for the tweet of the Example 1


Pythia
Enter text:
Our voices reveals lots of things, illnesses included #vtecl #m
Find Topics
Predicted Topics:
buy will insecure something BigData iPhone DataScience deserve nice phone literal ly nowthisisliving flip one break things IoT android billionairepre sident ancient

Health: 1.0
Culture: 0.62
Personal life: 0.62
Belief: 0.62
Humanities: 0.45

Fig. 3. Pythia system output for the tweet of the Example 10

REFERENCES

[1] "Wikipedia, the free encyclopedia." [Online]. Available: https://en.wikipedia.org/wiki/Main_Page

[2] M. Chen, X. Jin, and D. Shen, "Short text classification improved by learning multi-granularity topics," in *Proceedings of the 22nd IJCAI - Volume III*, ser. IJCAI'11, 2011, pp. 1776–1781.

[3] L. Yang, C. Li, Q. Ding, and L. Li, "Combining lexical and semantic features for short text classification," *Procedia Computer Science*, vol. 22, pp. 78 – 86, 2013.

[4] I. Litou, V. Kalogeraki, and D. Gunopulos, "On topic aware recommendation to increase popularity in microblogging services (short paper)," in *On the Move to Meaningful Internet Systems: OTM 2016 Conferences*, 2016, pp. 673–681.

[5] Q. Chen, T. Shipper, and L. Khan, "Tweets mining using Wikipedia and impurity cluster measurement," *2010 IEEE ISI*, pp. 141–143, 2010.

[6] "Api:main page - mediawiki." [Online]. Available: https://www.mediawiki.org/wiki/API:Main_page

[7] "Stanford corenlp - a suite of core nlp tools." [Online]. Available: http://stanfordnlp.github.io/CoreNLP/index.html

[8] "Stanford named entity recognizer (ner)." [Online]. Available: http://nlp.stanford.edu/software/CRF-NER.shtml

[9] J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ser. ACL '05, 2005, pp. 363–370.