Inspiratron.org - Natural language processing, machine learning and cybersecurity

Political bot (AI) fighting human bots (using NLP and OCR)

by Nikola Miloševi? - Sunday, March 02, 2014

https://inspiratron.org/blog/2014/03/02/political-bot-fighting-human-bots/

Probably I should write this on Serbian, but to keep consistency, English it is.

Since soon elections gonna be held in Serbia, there is a lot of talk about political campaigns. And one of the major issue in the news are human bots applied in the political campaign on the internet. Since parties in Serbia have too many members (it is estimated that almost every second person in the country is member of some party), they applied their members as bots to watch over news articles on internet portals and comment (make people vote for the party they are members of). Couple of years ago, there was no internet campaign at all in Serbia. Now, thousands of people are commenting articles on the internet for different political parties. Situation on job market is quite bad, so people in search for a job are becoming party members, hoping that if they do enough for a party, the party will help them find a job. Party bot's job is quite mindless, read the article and put some contextual, or even non contextual comments on it, under various user names and saying that they are from various backgrounds. What they often do is just copy-paste of some comment.

This is however, not just presumption, there is presentation of one party which were doing it last elections on slideshare, explaining how it was done. Here is the link. It is worth to say that author of presentation is now adviser of Serbian president.

So, enough of commenting situation in Serbia, back to the point. Since, this human bots, are obviously called bots... same job probably can do a machine (by using a bit of natural language processing - NLP and optical character recognition - OCR). So I decided to sit down yesterday evening, about 9pm, and try to write an application in Python, a prototype, that can show case that machine can do it. I have to make disclaimer, it was done in something like 4 hours, starting at 9pm evening, after my work, till about 1am morning. It is not perfect, but it can showcase. My code, if you want to play or modify can be found on github: https://github.com/nikolamilosevic86/politic-bot. Please do pull requests if you improved it.

How should it work?

So first question is how should it work. The idea is to crawl news article, extract relevant text. Read it and find protagonist (political party or political leader). Randomly select comment, put protagonist in it, and post it. Easy as this. Comments won't look too random, since application has a bit of intelligence to find protagonist of article. I have chosen to showcase application on B92.net portal, since it is one of the most read news portal and one with the best fraud detection (caption, moderation). Here will be shown, how all those were quite useless for this approach (they could block some of the comments, but with some improvements on captcha detection system, it won't be able to stop it at current state of system, and our comments would be considered as a normal human comments even by moderator).

Crawl article

I wanted to just showcase, so the application is taking input in array, in my case three articles. BeautifulSoup Python module is used for HTML parser. Here is some part of code:

```
import urllib2
#beautiful soup needs to be installed using apt-get install python-bs4
from bs4 import BeautifulSoup
from urlparse import urlparse
import gc
import re
from HTMLParser import HTMLParser
import operator
from random import randint
from PIL import Image
import ImageEnhance
from pytesser import *
from urllib import urlretrieve
import urllib

def get(link):
    urlretrieve(link,'temp.png')

class MLStripper(HTMLParser):
    def __init__(self):
        self.reset()
        self.fed = []
    def handle_data(self, d):
        self.fed.append(d)
    def get_data(self):
        return ''.join(self.fed)

def strip_tags(html):
    s = MLStripper()
    s.feed(html)
    return s.get_data()
stopwords = [line.strip() for line in open('stopwords')]
start_pages = ["http://www.b92.net/info/vesti/index.php?yyyy=2014&mm=02&dd=28&nav_category=11&
nav_id=818066", "http://www.b92.net/biz/vesti/srbija.php?yyyy=2014&mm=02&dd=28&nav_id=818018",
 "http://www.b92.net/info/vesti/index.php?yyyy=2014&mm=02&dd=28&nav_category=11&nav_id=817915"
]
for start_page in start_pages:
    f = urllib2.urlopen(start_page)
    parsed_uri = urlparse( start_page )
    domain = '{uri.scheme}://{uri.netloc}/'.format(uri=parsed_uri)
    print domain
    html = f.read()
    f.close()
    parsed_html = BeautifulSoup(html)
    comment_button = parsed_html.body.find('li', attrs={'class':'comment-send'})
    comment_link = domain[:-1] + comment_button.find('a')['href']
    article_text = parsed_html.body.find('div', attrs={'class':'article-text'})
    article_text = (re.subn(r'<(script).*?(?s)', '', str(article_text))[0])
    article_text = (re.subn(r'<(style).*?(?s)', '', str(article_text))[0])
    article_text = (re.subn(r'<(div) id="related-box".*?(?s)', '', str(article_text))[0])
    article_text = (re.subn(r'<(div) class="img-caption".*?(?s)', '', str(article_text))[0])
    article_text = (re.subn(r'<(div) id="social".*?(?s)', '', str(article_text))[0])
    article_text = (re.subn(r'Foto: Tanjug', '', str(article_text))[0])
    article_text = (re.subn(r'Tweet', '', str(article_text))[0])
    article_text = (re.subn(r'<(span) class="article-info2.*?(?s)', '', str(article_text))[0])
    article_text =  strip_tags(article_text)
    tokens = article_text.split(" ")
```

In this part of code is shown imports and some utility functions and class in the beginning. Stopwords are read from the file for later use, and articles are crawled. Page is parsed and article text part is found. Some parts of html page is stripped like script and style tags with its content. Also some divs with social media buttons, images are deleted, so we end up with just plain article text. Next article is split to tokens. Every token in our case is a word. Now we need to filter a stopwords. Stopwords are basically the words that are very frequent and does not carry much of the meaning. They are quite used in natural language processing, and especially in approaches like bug-of-words. We can consider our somehow like bag-of-words. If tokens are not members of stopwords, we create hash map with tokens as keys and frequency of occurrence as values. Here is the code:

```
tokens2 = {}
    for token in tokens:
        if(token not in stopwords):
            if(token not in tokens2):
                tokens2[token] = 1
            else:
                tokens2[token]= tokens2[token]+1
```

Now we sort our hash map by token occurrences, and we can get among top 10 tokens our protagonist. However, system has to know who usual protagonists are, so a dictionary is created for political parties names and for political leader names. A simple Name Entity Recognition lookup is done on the top 10 tokens and ether political leader or party is selected. For simplicity we made a dictionary for comments, with couple of comments that are randomly selected and the protagonist is inserted in. They have following format:

```
[Leader] je najveci car na svetu. Sta je on uradio za ovo kratko vreme, ne moze da se meri sa
ostalima!
[Party] ce nam zemlju pretvoriti u utopiju. Nista nam nece biti ravno.
```

[Leader] and [Party] tags are replaced with the selected protagonist. This can be more modified and made more robust using some dependency grammars, multiple sentence dictionaries, sentiment analysis etc. But here is point to showcase, not to make complex system (complex system cost lot of effort and probably a some amount of money). Here is the code for described part.

```
 sorted_tokens = sorted(tokens2.iteritems(), key=operator.itemgetter(1),reverse=True)

    for x in range(0,10):
        print sorted_tokens[x]

    Leaders = [line.strip() for line in open('Leaders')]
    Parties = [line.strip() for line in open('Parties')]
    Unames = [line.strip() for line in open('UserNames')]
    LeaderComments = [line.strip() for line in open('LeaderComments')]
    PartyComments = [line.strip() for line in open('PartyComments')]
    leader = ''
    party = ''
    for x in range(0,10):
        if sorted_tokens[x][0] in Leaders:
            leader = sorted_tokens[x][0]
            break
    if leader == '':
        for x in range(0,10):
            if sorted_tokens[x][0] in Parties:
                party = sorted_tokens[x][0]
                break
    print "Leader = " + leader
    print "Party = " + party
    comment = ''
    if leader != '':
        comment = LeaderComments[randint(0,len(LeaderComments)-1)].replace('[Leader]', leader)
    else:
        comment = PartyComments[randint(0,len(PartyComments)-1)].replace('[Party]', party)
```

```
    print comment
```

As you can see, comments are selected randomly from the list. Now we need to post the comment that was created using this fancy Natural language processing (NLP) technique. We need to make request of some populated form. Which is not a problem in most cases. But, there is a captcha!

Dealing with CAPTCHA

A CAPTCHA is a type of challenge-response test used in computing as an attempt to ensure that the response is generated by a person. The process usually involves one computer (a server) asking a user to complete a simple test which the computer is able to generate and grade.The term "CAPTCHA" was coined in 2000 by Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford (all of Carnegie Mellon University). It is an acronym based on the word "capture" and standing for "Completely Automated Public Turing test to tell Computers and Humans Apart".

CAPTCHA is designed to be solvable only by humans, but however, it is not always the case. Approach to solve captcha is similar to human approach, optically recognize characters on image. How it can be done - by using some OCR engine (Optical Character Recognition). In my case, I used pytesser library for python, witch is using tesser OCR engine. It uses tesser as external tool, putting image in, and taking output text. I had some problems installing Tesser, Leptonica and all it's dependencies. It took me for about hour, so it can be done. Now, image have to be a bit clearer. Normal captcha image from B92 looks like this:

We need to make it more readable and remove background noise. This is done by heuristics. the colours in background are quite bright, so we used logic that background colours have RGBA parameters higher then 120. At the end of filtering we got this:

Now in many cases Tesser is able to recognize characters. However, maybe tesser is not the best engine, and maybe some more heuristics could be used to improve performances, but in many cases tesser did recognized all the character correctly.

Here is the code for OCR part:

```
  get(capcha_image);
    im = Image.open("temp.png")
    nx, ny = im.size
    im2 = im.resize((int(nx*5), int(ny*5)), Image.BICUBIC)
    im2.save("temp2.png")
    enh = ImageEnhance.Contrast(im)
    enh.enhance(1.3)#.show("30% more contrast")
    imgx = Image.open('temp2.png')
    imgx = imgx.convert("RGBA")
    pix = imgx.load()
    for y in xrange(imgx.size[1]):
        for x in xrange(imgx.size[0]):
            if pix[x, y][0]>120 and pix[x, y][1]>120 and pix[x, y][2]>120 and pix[x, y][3]>120
:
                pix[x, y] = (255, 255, 255, 255)
            else:
```

```
                pix[x, y] = (0, 0, 0, 0)
    imgx.save("bw.gif", "GIF")
    original = Image.open('bw.gif')
    #bg = original.resize((231, 72), Image.NEAREST)
    bg = original.resize((198, 72), Image.NEAREST)
    ext = ".tif"
    bg.save("input-NEAREST" + ext)
    image = Image.open('input-NEAREST.tif')
    captcha_string =  image_to_string(image).replace(" ","").replace("\n","").replace("%","8")
.replace('\N','W').replace('l/','V')
    if captcha_string == '':
        bg = original.resize((500, 241), Image.NEAREST)
        ext = ".tif"
        bg.save("input-NEAREST" + ext)
        image = Image.open('input-NEAREST.tif')
        captcha_string =  image_to_string(image).replace(" ","").replace("\n","").replace("%",
"8").replace('\N','W').replace('l/','V')

    print captcha_string
```

Now it only remains to send requests. This part I will not explain in details, just that form is sent using POST request. You can find code on github.

Results

From several runs on some OCR recognized captcha well, and here are some images:

All three comments on this page are from the bot. Unfortunately for 2 comments bot didn't like this party, but on one it did. The comments are taken from library on totally random way and they does not reflect opinion of the author of this blog (myself). So please, if you are supporter of some party my bot didn't like, do not feel offended.


PETAK 28.02.2014 | 20:31

"Politika DS može da se dodirne"

IZVOR: B92

Požarevac -- Predsednik DS Dragan Đilas pozvao je građane da im poverenje jer politika DS može da se dodirne i da se opipa, dok SNS nudi samo obećanja.

▸ Prikazati ovde celokupan tekst članka

POŠALJITE KOMENTAR

Dodaj oglas

Komentari | Hronološki | Preporučeni | Nepreporučeni

Ä€ilas ce promeniti sve na bolje. Zivecemo u utopiji kad on dodje na vlast.
(Otac75, 1. mart 2014 13:17)

Link komentara

Preporučujem (0) Ne preporučujem (0) ? Šta je ovo

Iskeno mi je jasno sta Dragan Djilas pokusava da postigne svojim izjavana ali ja njegov rad u Beogradu vidim iskljucivo kao reklamu. Kupljeni autobusi plavo zute boje koji se voze iskljucivo po centru grada. Sve sto moze da se opipa u Beogradu je gradjeno od novca gradjana Beograda. Licemerne su izjave ovog tima i mislim da ce doneti samo stetu Djilasu i DS-u.
(BogdanBGD, 1. mart 2014 13:13)

Here it can be seen that it has problems with special characters, but even thou protagonist has special character in name, he was recognized well. I even got thumbs up on couple of bot comments.

So I believe that this can showcase that one computer with simple application (my prototype had 155 lines of python code, written with all the required research and installation efforts for about 4 hours) can be able to simulate and even beat the army of human bots that parties applies in political campaigns. Please stop using humans this way... it is inhuman.

All rights reserved and copyrighted by inspiratron.org and Nikola Milosevic