Université Paris Saclay

OPT 7

Deep learning for structured data

Gender classification

Authors:
Issa Hammoud
Salim Tabarani
Youcef Madji

Supervisor:
Prof. Alexandre Allauzen

March 4, 2019

Contents

Introduction

Deep learning had shown an impressive results in the past few years on different problems and tasks, from vision to speech and natural language processing. Hence, deep learning approaches have replaced shallow methods, but not in all problems.
The problem of gender classification have an interest in some domains, like marketing for ad targeting, or to perform some statistics for literature purposes, etc... In general, it is a need for author profiling.
However, the classification's difficulty depends a lot on the data. For instance, classifying novels on gender is very different than classifying blog posts.
In this work, we present several methods that address the problem of gender classification on blog posts data. In the first chapter, we discover our data and we calculate a baseline to estimate the task difficulty. Then, we begin with 3 solutions that are as follows:
In the second chapter, we present a method based on a combination of recurrent and convolutional neural networks, in the third chapter, we present a shallow method based on human designed features and in the forth chapter we present another method based on deep learning. Finally, we propose some additional ingredients that can be relevant and increase our score.

1 Data exploration and baseline

1.1 Data exploration

First of all, we should explore and prepare our data in order to understand them well. The corpus consists of collected posts of 19,320 bloggers gathered from blogger.com in August 2004. Each blog is presented as a separate file, the name of which indicates a blogger id and the blogger's self-provided gender, age, industry and astrological sign.
Each file contains mainly 3 tags : $< Blog >, < date >, < post >$. We extracted their contents using regular expression, and we separated each post individually, so in our data frame, each person appears as many as he had posts.
Precisely, the corpus incorporates a total of 681,288 posts overall, which implies 35 posts per person in average. There are equal number of posts per gender (50.67% for male, 49.33% for female)
Below are some useful statistics calculated after eliminating English stop words:

words number	**unique words number**	**max post length**	**min post length**	**average post len**	**median post len**	**80% of posts length $\leq$**
101,264,977	3,912,652	78,477	2	121	71	180

Table 1.1: Data statistics (without English stop words)

	id	gender	age	industry	astrologic	post	date
0	4200843	male	39	Technology	Sagittarius	\n\n \n Why suns , obsession trying l...	21,August,2004
1	4200843	male	39	Technology	Sagittarius	\n\n \n I'ordered Counter Strike:CZ ...	18,August,2004
2	4200843	male	39	Technology	Sagittarius	\n\n \n Remember jokes? Well I I nev...	18,August,2004
3	4200843	male	39	Technology	Sagittarius	\n\n \n If 'watch 'Waking Dead' make...	16,August,2004
4	4200843	male	39	Technology	Sagittarius	\n\n \n Sunday always feels like Sun...	15,August,2004

Figure 1.1: Head of the data frame

1.2 Baseline

Before investigating in strong methods that are used for text classification, we should have an idea on the difficulty of the problem. Thus, we tried baseline methods based on the use of word embeddings with one fully connected layer of 5 neurons on top of it.
The 3 methods differ only in the embedding layer. The first one used a randomly initialized embedding layer, while the second and third one use pretrained embeddings, set to non trainable and trainable respectively.

	Random embeddings	**Pretrained static embeddings**	**Pretrained trainable embeddings**
accuracy	69.2%	56%	69.1%

Table 1.2: Baseline results

These results are obtained using only the posts. The parameters that gave the best result for random embeddings are summarized in the table below:

Embedding dimension	**max length**	**max words**	**dropout**
300	180	50000	0.5

Table 1.3: Best parameters

We can remark from the results that the problem is not easy, and it worths to investigate in more advanced methods.

2 Recurrent Convolutional Neural Network(RCNN)

2.1 Introduction

Recurrent Neural Network (RNN) represents nowadays a standard approach for applications applied to text. It has the advantage of capturing the structure in a text, that takes in consideration the words and sentences order. One of the advanced RNN is the Long Short Term Memory (LSTM) that captures long dependencies in order to solve the vanishing gradient problem. However, it is quite costly especially with long texts.
In the other hand, convolutional neural network (CNN) was found to perform well, not only on image data, but also on texts. The same properties that make CNN excellent at computer vision also make them highly relevant to sequence processing. Time can be treated as a spatial dimension, like the height or width of a 2D image. Such 1D convnets can be competitive with RNN on certain sequence-processing problems, usually at a considerably cheaper computational cost.
This approach, first introduced by Lai *et al.*, combines the two mentioned models. It applies a recurrent structure to capture contextual information as far as possible when learning word representations, which may introduce considerably less noise compared to traditional window-based neural networks. It also employs a max-pooling layer that automatically judges which words play key roles in text classification to capture the key components in texts.

2.2 RCNN model

The model combines a word and its context to present a word. The context helps to obtain a more precise word meaning. In our model, we use a recurrent structure, which is a bidirectional recurrent neural network, to capture the context.
We define $c_l(w_i)$ as the left context and $c_r(w_i)$ as the right context of word w_i. These quantities as calculated as follows:

$$c_l(w_i) = f(W^{(l)}c_l(w_{i-1}) + W^{(sl)}e(w_{i-1}))$$
$$c_r(w_i) = f(W^{(r)}c_r(w_{i+1}) + W^{(sr)}e(w_{i+1}))$$

Where :

- $W^{(l)}$ is a matrix that transforms the hidden layer (context) into the next hidden layer.

- $W^{(sl)}$ is a matrix that is used to combine the semantic of the current word with the next word's left context.
- $e(w_{i-1})$ is the word embedding of word w_{i-1}.

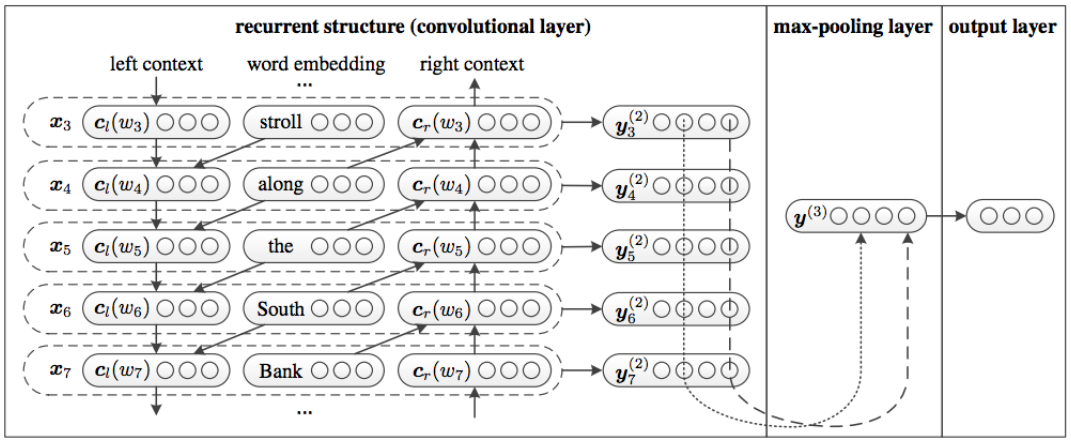

Figure 2.1: RCNN structure

In this manner, using this contextual information, our model may be better able to disambiguate the meaning of a word w_i compared to conventional neural models that only use a fixed window (i.e., they only use partial information about texts). So, each input x_i is a concatenation between $e(w_i)$, $c_l(w_i)$ and $c_r(w_i)$.

$$x_i = [c_l(w_i); e(w_i); c_r(w_i)]$$

Now we can apply a linear transformation together with the tanh activation function to x_i and send the result to the next layer.

$$y_i^{(2)} = tanh(W^{(2)}x_i + b^{(2)})$$

$y_i^{(2)}$ is a latent semantic vector, in which each semantic factor will be analyzed to determine the most useful factor for representing the text. Then, the max pooling step can take place. The k-th element of $y_i^{(3)}$ is the maximum in the k-th elements of $y_i^{(2)}$.
The pooling layer converts texts with various lengths into a fixed-length vector. With the pooling layer, we can capture the information throughout the entire text. The last part of the model is an output layer. Similar to traditional neural networks, it is defined as:

$$y^{(4)} = W^{(4)}y^{(3)} + b^{(4)}$$

Finally, a softmax or a sigmoid can be applied on top of the model.

2.3 Results

Our implementation differs a bit with the described architecture.
In fact, instead of using a linear transformation after the concatenation, we used a one-dimensional convolutional layer, in order to catch most of the relevant features.In addition, we initialized the first and last elements of the left and right contexts, respectively, with zeros instead of what is proposed in the paper.
Our hyper parameters are as follows:

- LSTM dimension of 200.
- 1 kernel conv1d of dimension 100.
- Optimizer Adadelta.
- batches of size 256.

These values are handy chosen, and we didn't have the possibilty to try another combination because it takes a lot of time. The final score was about 70% of accuracy.

3 Second approach: human-designed features

3.1 F-measure

F-measure explores the notion of implicitness of text and is a unitary measure of text's relative contextuality.

F-measure is defined based on the frequency of the POS usage in a text:

$F = 0.5 * [(freq.noun + freq.adj + freq.prep + freq.art)–(freq.pron + freq.verb + freq.adv + freq.int) + 100]$

3.2 Gender Preferential Features

This features are inspired from an email gender classification task.The language expressed by males is more proactive at solving problems while the language used by females is more reactive to the contribution of others.

So we will use the following 10 features that are based on the presence of suffixes and apologies in adjectives and adverbs:

f1	words ending with able
f2	words ending with al
f3	words ending with ful
f4	words ending with ible
f5	words ending with ic
f6	words ending with ive
f7	words ending with less
f8	words ending with ly
f9	words ending with ous
f10	sorry words

Table 3.1: Gender preferential features

3.3 Surface features

We will now focus on the morphology of the posts by using the following surface features:

1. Number of sentences in a post

2. Number of words in a post
3. Number of characters in a post
4. average of the number of alphabets in a sentence
5. average of the number of digits in a sentence
6. Number of special characters in a post
7. Number of punctuation marks in a post
8. number of short words in a post(less than 4 characters)
9. average word length
10. Average sentiment score of the post
11. lexical richness based on Yule's K index

3.4 POS Sequence Pattern Features

Using this word class features, we calculate the number of words in a post that belong to each of these classes.

female-biased	*shopping, mom, cried, freaked, pink, cute, gosh, kisses, yummy, mommy, boyfriend, skirt, adorable, husband, hubby*
male-biased	*linux, microsoft, gaming, server, software, gb, programming, google, data, graphics, india, nations, democracy, users, economic*
acronyms	*gm, gn, bbl, rofl, afaik, lmao, lmfao, lol, tgif, omg, omfg, tc, ttyl, af, ama, bae, dm, dafuq, eli5, fml, fify, hifw, icymi, imho, irl, mrw, yolo, xoxo*
chastity	*sexual, celibacy, abstinence, virtue, monogamy, virginity, pureness, unchaste, fasting, frugality, self-restraint, abnegation, asceticism, avoidance, renunciation, self-control, sobriety, teetotalism, monasticism, austerity, abstain, diet, hungry*
predictions	*forecast, guess, prognosis, prophecy, augury, conjecture, divination, foretell, hunch, horoscope, omen, presage, prevision, zodiac, fortune-telling, surmising*
envy	*hatred, malice, prejudice, rivalry, backbiting, coveting, grudge, heartburn, malevolence, opposition, green-eyed, invidious, resentful*
rain	*deluge, drizzle, flood, hail, mist, monsoon, precipitation, rainfall, shower, sleet, stream, torrent, pour, spate, spit, sprinkle, wet-stuff*
evildoer	*criminal, devil, felon, gangster, lawbreaker, murderer, psychopath, sinner, sociopath, troublemaker, villain*
mathematics	*mathematics, algebra, arithmetic, geometry, trigonometry, calculus, math, addition, division, multiply, figures, pie, chart, numbers, subtraction, plot, graph, data*
groom	*groom, bride, suitor, benedict, spouse, fiancé*
ethics	*ethics, belief, conduct, convention, honesty, honor, mores, values, ethos, practice, principles, conscience*
fear	*fainthearted, terrified, angst, anxiety, concern, despair, dread, horror, jitters, panic, unease, worry, agitation, aversion, awe, consternation, creeps, discomposure, disquietude, distress, fright, nightmare, phobia*
disgust	*antipathy, dislike, distaste, loathe, revulsion, abhorrence, abominate, detest, hateful, nausea, objection, repugnance, revolt, satiation, satiety, sickness, surfeit*

Figure 3.1: Word class features

3.5 Test

We now need to test the performance of these features for the gender classifciation of blog posts. We obtain the following result on the test set:

Model	Accuracy
XGBOOST	60.4%

Table 3.2: Model accuracy

4 CNN Long Short-Term Memory Networks

4.1 Introduction

The CNN LSTM architecture involves using Convolutional Neural Network (CNN) layers for feature extraction on input data combined with LSTMs to support sequence prediction. A simple CNN-LSTM model combination consists of an initial convolution layer which will receive word embeddings as input. Its output will then be pooled to a smaller dimension which is then fed into an LSTM layer. The intuition behind this model is that the convolution layer will extract local features and the LSTM layer will then be able to use the ordering of said features to learn about the input's text ordering.

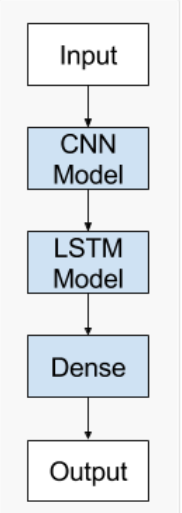

Figure 4.1: Basic Convolutional Neural Network Long Short-Term Memory Network Architecture

4.2 CNN-LSTM model

For this architecture, we draw inspiration from the work done by Aleksandr Sboev *et al.*. It consists of an initial convolution layer which will receive word embeddings as input. Its output will then be pooled to a smaller dimension with a dropout layer. The 3 layers will be repeated 3 times the result with then fed into an LSTM layer.

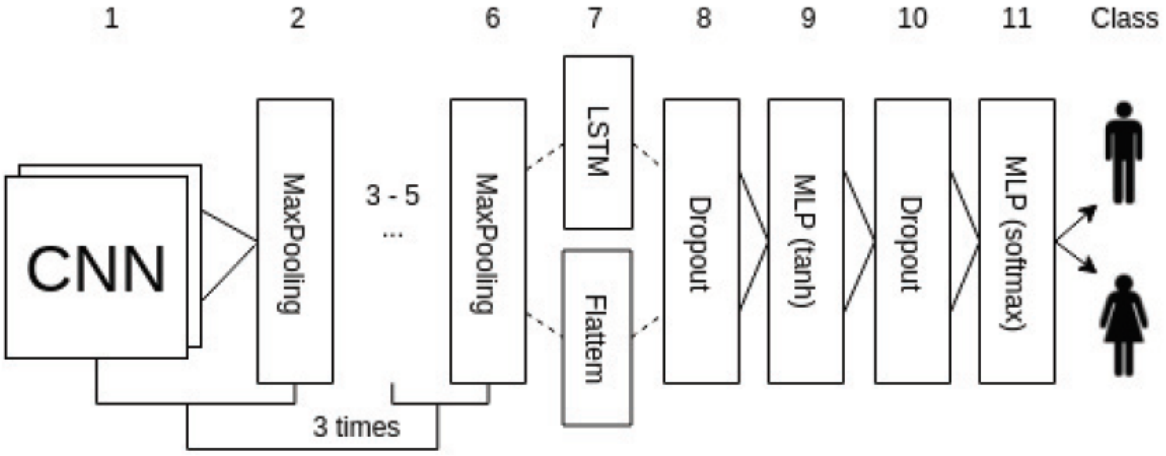

Figure 4.2: Neural network structure with combination of CNN, MLP and LSTM

A complicated neural network combining CNN, MLP and LSTM includes:

- Input layer is an embedding layer: we tried to train our own embeddings than we tried Glove and fastTest pre-trained embeddings.
- 1st, 3rd, 5th CNN layers: Number of convolution kernels to use = 128, the extension of each filter = 7, activation function is ReLU
- 2nd, 4th, 6th layers: MaxPooling (pool length = 2) + dropout(Fraction of the input units to drop = 0.3)
- 7th layer: Long-Short Term Memory (output dimension = 100)
- 8th layer: dropout layer. (Fraction of the input units to drop = 0.35) sentence
- 9th layer: fully connected NN layer (Number of hidden neurons = 128, activation function = relu)
- 10th layer: dropout layer (Fraction of the input units to drop = 0.45)
- 11th layer: fully connected NN layer (Number of hidden neurons = 1, activation function = sigmoid)

4.3 Test

We now need to test the performance of the model for the gender classifciation of blog posts. We obtain the following result on the test set:

Model	Accuracy
CNN-LSTM	70.92%
CNN-LSTM with Glove pré-trained embedding	71.46%
CNN-LSTM with fasetText pré-trained embedding	72.19%

Table 4.1: CNN-LSTM Model accuracy

5 Conclusion and perspective

The problem of gender classification was discussed in this work. We presented several methods that exist in the literature, 2 of them are based on deep learning and the other on human designed features.
As we can see in the results, the problem is not quite easy and need to be more studied in order to have acceptable accuracy, so we propose additional below additional steps that can potentially help:

- Try WRCNN, a variant of RCNN, that can be more suitable for this problem.
- Add the additional information in the corpus to the embedding vector.
- Try attention mechanism to get rid of the long length sentences.