FAN FICTION GENRE CLASSIFICATION

Statistical Learning Final Project

Sapienza University of Rome, Master's Degree in Data Science

Authors:
Diletta Abbonato, Laura Laurenti,
Anna Presciuttini, Santo Palaia

July, 2020

Contents

1 Abstract

Text classification is the process of assigning categories to documents according to their content. We apply different learning techniques in order to perform a fan fiction genre classification. In its simplest form, fanfiction is when somebody takes a character, universe, or story from a different scenario to create their own story. These characters and scenarios are commonly pulled from novels, TV shows, movies, and even real life. We chose for our work 5 genres that are: Fantasy, Romance, Horror, Science Fiction, Humor. Neural Networks (NNs) were originally designed for deep learning computer vision tasks, but they have proven highly useful for Natural Language Processing (NLP) tasks as well.

2 Main research aim and framework

The main goal is to detect the genre of the fanfiction. The idea come from our teenager memories where we were obsessed with famous characters as Harry Potter and Harry Styles and our immagination navigated the ship of our tender heart. To implement our idea we were inspired by the paper made by a team of Nanyang Technological University which the topic is to detect the 5 big personal traits from a text.

3 The dataset

3.1 Data collection

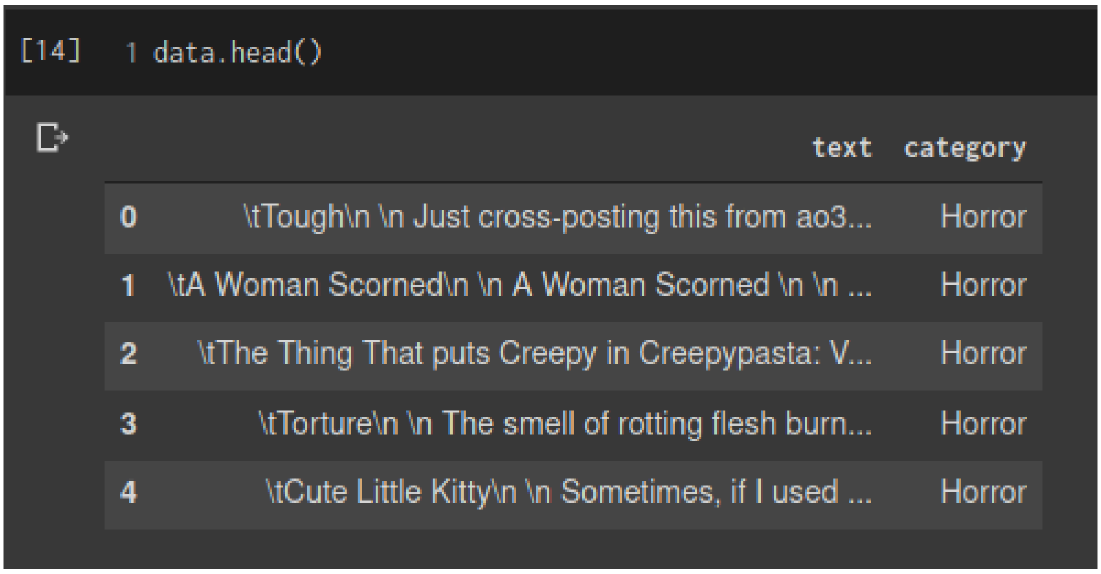
Data for this project come from fanfiction.net, a popular hub hosting, a large repository of freely available fan written content. We computed the data collection process scraping this fanfiction website through the use of BeautifulSoup, considering just english fan fiction.

```python
def soupify(url, rate_limit=3):

    from bs4 import BeautifulSoup as bs
    import requests
    import time

    time.sleep(rate_limit)

    html = requests.get(url).text
    return bs(html, 'html.parser')

genres=['Romance','Adventure','Sci-Fi','Horror','Fantasy']

scraper = Scraper()
i=0
meta=[]
story_m=[]
for i in tqdm_notebook(range(0,1157)):

    s=random.randint(0,980000)
    soup = soupify('https://www.fanfiction.net/s/' + str(i),
                   rate_limit=3)
    try:
        metadata = scraper.scrape_story_metadata(s)
        story = soup.find_all('div', {'class': 'xcontrast_txt'})
        story_text = story[1].text
        meta.append(metadata)
        story_m.append(story_text)

        if meta[i]['lang']!='English':
            del meta[i]
            del story_m[i]
            print('I\'m Delete :()')
    except:
        continue
```

At the end of the scraping, we save all the fanfictions correctly downloaded in a csv file, with only two columns: genre (of the fanfiction) and text (the story of the fanfiction).

In order to avoid genres unbalance, we save only 426 fanfiction (to do the minority class present in the scraped files) for each of the 5 genre analyzed for a total of 3945 fanfictions.

At the end of the scraping, the data set appear like this [1]:



```
[14]  data.head()
```

	text	category
0	\tTough\n \n Just cross-posting this from ao3...	Horror
1	\tA Woman Scorned\n \n A Woman Scorned \n \n ...	Horror
2	\tThe Thing That puts Creepy in Creepypasta: V...	Horror
3	\tTorture\n \n The smell of rotting flesh burn...	Horror
4	\tCute Little Kitty\n \n Sometimes, if I used ...	Horror

Figure 1: Start Data Set of the fanfictions

3.2 Data preprocessing

There are four significant steps in our data preprocessing:

- **De-capitalization**: we performed case-folding reducing everything to lower case. This because, for example, our model might treat a word, which is in the beginning of a sentence with a capital letter different from the same word, which appears later in the sentence but without any capital letter. This might lead to decline in the accuracy.

- **Puntuaction and non-textual characters removal**: We replaced puntuaction and non textual-characters splitting by whitespaces in order to perform a better tokenization.
- **Tokenization and stopwords removal**: We segmented text into sentences and we removed common words that generally do not contribute to the meaning of a sentence for the purposes of information retrieval.
- **Parts of speech tagging**: We marked up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context.

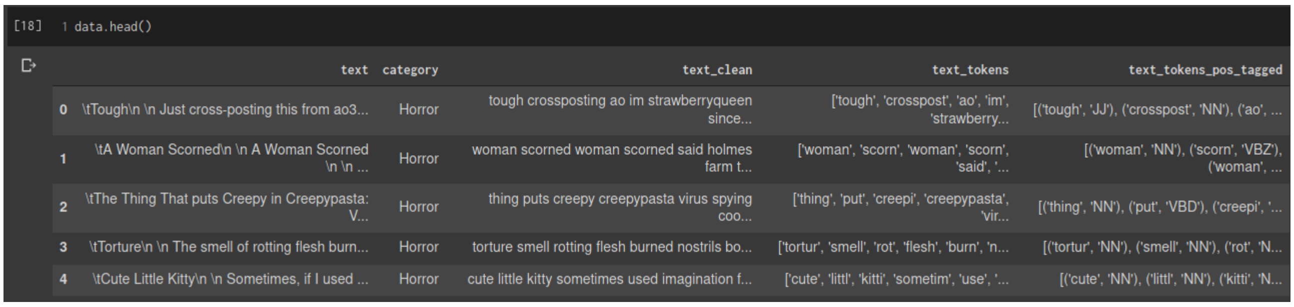
```
def  clean_text(df, text_field, new_text_field_name):
  df[new_text_field_name] = df[text_field].str.lower()
  df[new_text_field_name] = df[new_text_field_name].apply
  (lambda elem: re.sub(r"(@[A-Za-z0-9]+)
  |([^0-9A-Za-z \t])|(\w+:\/\/\S+)|
  ^rt|http.+?", "", elem))
  # remove numbers
  df[new_text_field_name] = df[new_text_field_name].apply
  (lambda elem: re.sub(r"\d+", "", elem))

  return df

def word_pos_tagger(text):
    pos_tagged_text = nltk.pos_tag(text)
    return pos_tagged_text

data = clean_text(data, 'text', 'text_clean')
```

As the code below shows, we applied nltk library in order to correctly remove stopwords and perform tokenization:

```
from nltk.corpus import stopwords
stop = stopwords.words('english')
stop.append('\t')
data['text_clean'] = data['text_clean'].apply(lambda x: ' '.join([word for
word in x.split() if word not in (stop)]))



nltk.download('punkt')
ps = PorterStemmer()
data['text_tokens'] = data['text_clean'].
apply(lambda x:
word_tokenize(x)) # tokenization
data['text_tokens'] = data['text_tokens'].
```

```
apply(lambda x:
[ps.stem(y) for y in x]) # stemmization
```

At the end of the preprocessing, we obtained a dataframe with the following columns: category, text, *text_clean*(text without any punctuation), *text_tokens*(tokenization of the *text_clean*) and *text_token_POS* (tuples of token and tag for each token in *text_tokens*), as we can see in figure 2.



```
[18] data.head()
```

	text	category	text_clean	text_tokens	text_tokens_pos_tagged
0	\tTough\n \n Just cross-posting this from ao3...	Horror	tough crossposting ao im strawberryqueen since...	['tough', 'crosspost', 'ao', 'im', 'strawberry...	[('tough', 'JJ'), ('crosspost', 'NN'), ('ao', ...
1	\tA Woman Scorned\n \n A Woman Scorned \n \n ...	Horror	woman scorned woman scorned said holmes farm t...	['woman', 'scorn', 'woman', 'scorn', 'said', '...	[('woman', 'NN'), ('scorn', 'VBZ'), ('woman', ...
2	\tThe Thing That puts Creepy in Creepypasta: V...	Horror	thing puts creepy creepypasta virus spying coo...	['thing', 'put', 'creepi', 'creepypasta', 'vir...	[('thing', 'NN'), ('put', 'VBD'), ('creepi', '...
3	\tTorture\n \n The smell of rotting flesh burn...	Horror	torture smell rotting flesh burned nostrils bo...	['tortur', 'smell', 'rot', 'flesh', 'burn', 'n...	[('tortur', 'NN'), ('smell', 'NN'), ('rot', 'N...
4	\tCute Little Kitty\n \n Sometimes, if I used ...	Horror	cute little kitty sometimes used imagination f...	['cute', 'littl', 'kitti', 'sometim', 'use', '...	[('cute', 'NN'), ('littl', 'NN'), ('kitti', 'N...

Figure 2: Data Set after preprocessing

4 Models and Methods

We tested 4 non-deep learning models, whose accuracy levels were compared with a Neural Network model. Regarding the non deep learning models we have chosen:: Random Forest, Linear Support Vector Machine Classifier, Naive Bayes Classifier and Logistic Regression. After a first analysis, Linear Support vector machine is the best among non-deep learning methods. (see Appendix A).

4.1 Neural Network Architecture

A Neural Network is a mathematical model built by artificial neurons that get inspired by a biological neural network. We can consider neural networks as an algorithm used to solve hard coding-complex problems and they are a cornerstone of the learning machine today. We can consider a neural network as a model where we have an input (data input), one or more intermediate layers where we process this data, an output layer that is the final result. A neural network is made up of units called neurons, where each neuron is typically connected to all the neurons of the next layer of weighted connections. Each neuron adds the weighted values of all the now connected neurons and adds a bias value. An "activation function" is applied to this result, which transforms the value before passing it on to the next layer. In this way the input values are propagated through the network to the output values. In synthesis what happens inside these layers is to regulate weights and bias through different techniques in order to get to the desired result as we will tell later on.

4.1.1 Activation function

In our case we used non-linear functions as activation function because they help the model to adapt to different types of data. In particular the activation function used for the input and the hidden layer was the hyperbolic tangent activation function (tanh), a function very similar to sigmoid logistics with the difference that the range of values varies between -1 and 1

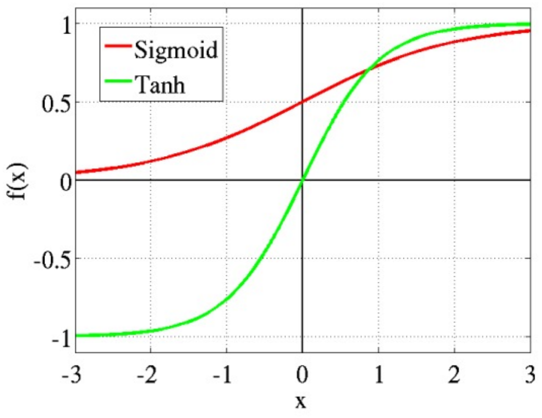


Figure 3: Tanh function

The advantage of using this function is that negative inputs will be strongly mapped as negative and instead zero inputs will be mapped close to zero. In addition, the function is dfferentiable and is suitable for classification as in our case. As far as the output function is concerned, we have used the softmax, a generalization of logistic regression that can be used for multiclassification. It is a function that transforms a vector of k real values into a vector of k real values that add up to 1 so that the results can be interpreted as probabilities.

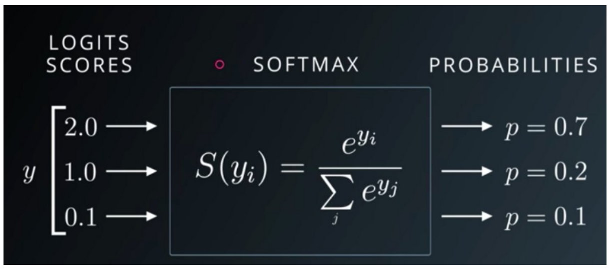


Figure 4: Softmax function

4.1.2 Optimization methods

Neural networks are difficult to train because inputs from previous layers can change after the weights are updated. For this reason we use optimization methods that allow us to optimize this learning, between these techniques we identify the batch normalization and dropout.

- **Batch Normalization**: The batch normalization is a simple heuristic method that allowed to train deep networks significantly better. It allows each layer to control the mean and the variance of its output, by appropriately rescaling them with a simple affine transformation.

- **Dropout**: The dropout is different from other layers because it is removed when not training. Consider the output from a generic layer $g(x)$ of the network, with dropout, during training, we replace it with

  $$\tilde{g}(x) = g(x) \odot m$$

  where m is a binary vector with entries drawn from a Bernoulli distribution with probability p.
  To do this, when not training we replace the output of the layer with its expected value during training:

  $$E(\tilde{g}(x)) = p * g(x)$$

  Failing to do so we introduce an undesired bias in the network behaviour.

- **Stochastic Gradient Descent Optimization**: The goal is to compute the gradient of the cost function; To do that, instead of an exact computation, we can do an approximation of the gradient using a smaller amounts of data. This technique is called Stochastic Gradient Descent. In origin, we want to train through the use of an expectation with respect to all data θ:

  $$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x^i))^2$$

  but, to limit the complexity we can use only a **mini-batch** B of M examples of the full data set, approximating the expectation with:

  $$J(\theta) \approx \tilde{J}(\theta) = \frac{1}{M}\sum_{i\in B}(y_i - f(x^i))^2$$

  In this case the computational complexity is fixed, give B and dose not depend of the size of the data set. Since we can assume that the samples are i.i.d, the Stochastic GD converges to a stationary point in average.
  In practise we can:

1. shuffle the data set
2. split the data set in block of size M and process them sequentially
3. after the last block, return to point 1 and iterate

A cycle of the previous steps are called **epoch**.

The choice of B is not straightforward:

- Smaller imply speed but require more iterations (due to noise);
- Larger imply precision but each iteration is more slower.

4.2 Summary

The following scheme shows a summary of the main points of our neural network, or better of the layers inside our NN 5.

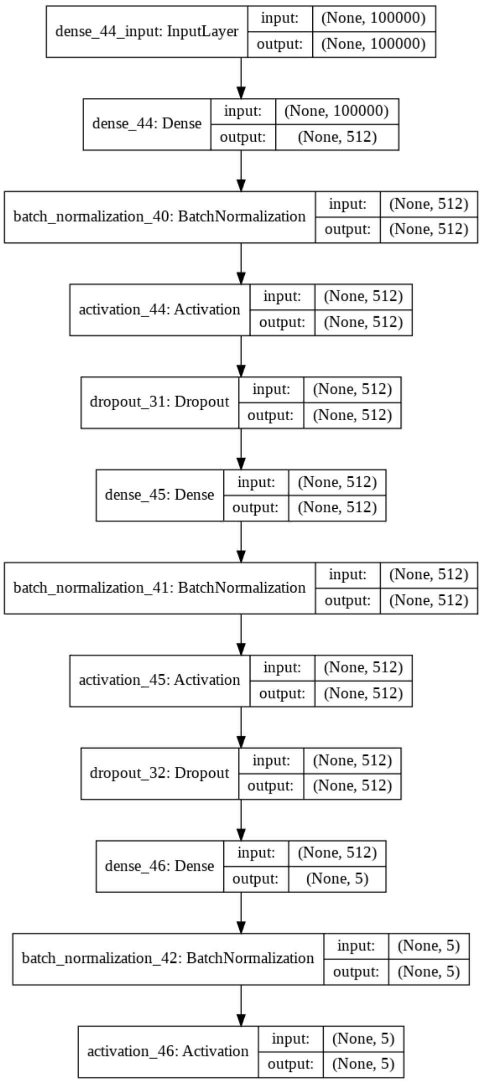


Figure 5: NN Model

5 Results

As we can see in figure[6], the change in the epoch as regards the loss of the model, we have that both for the train and for the valley there is a decrease in the value of the loss function. This decrease is much less stable in validation. As for the accuracy of the model, the train has a stationary point (100% of the acuracy) while for the val we get a growth. also in this case, we obtain a more unstable growth.

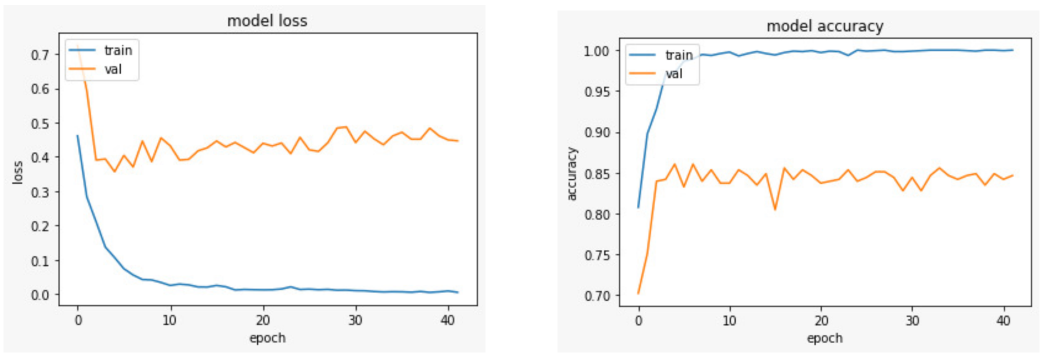


Figure 6: 2 Model loss and Model accuracy

About the confusion matrix [7] we can see:

- on the diagonal: the correctly classified;
- on the upper and lower triangular the improperly classified.

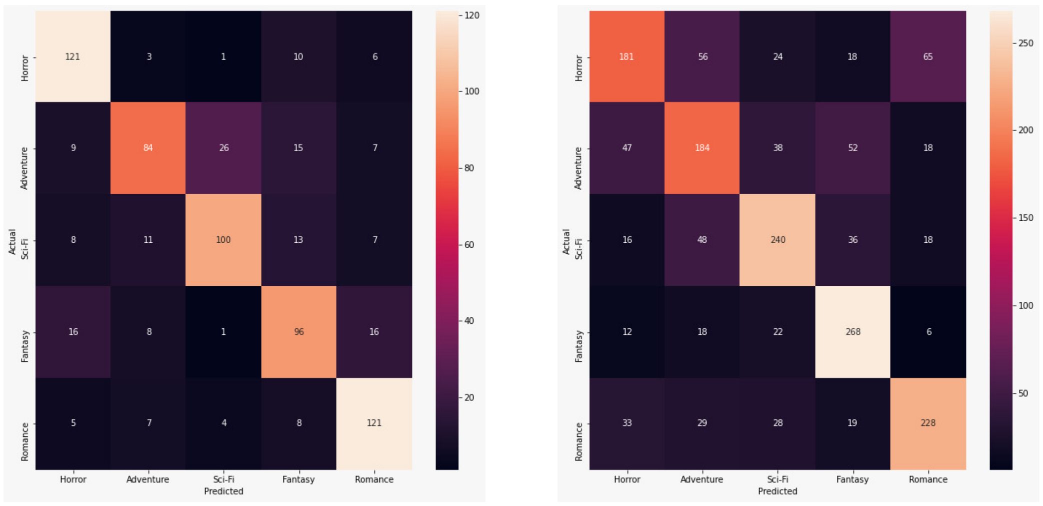


Figure 7: Confusion matrix for LSCV and NN

6 Software and Hardware toolkit

This section lists all of the Python packages and modules we used.

- Pandas: Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive.
- BeautifulSoup: Beautiful Soup is a library that makes it easy to scrape information from web pages. It sits atop an HTML or XML parser, providing Pythonic idioms for iterating, searching, and modifying the parse tree.

- NLTK: NLTK stands for Natural Language Toolkit. This toolkit is one of the most powerful NLP libraries which contains packages to make machines understand human language and reply to it with an appropriate response. Tokenization, Stemming, Lemmatization, Punctuation, Character count, word count are some of these packages
- Keras: Keras is a free open source Python library for developing and evaluating deep learning models. It wraps the efficient numerical computation libraries Theano and Tensor Flow and allows you to define and train neural network models

7 Appendix A

7.1 Logistic regression

Logistic regression is a regression model. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as “1”. Just like Linear regression assumes that the data follows a linear function, Logistic regression models the data using the sigmoid function

$$g(x) = \frac{1}{1 + e^{-x}}$$

In our case we have a multinomial Logistic regression: target variable have 3 or more possible types which are not ordered(i.e. types have no quantitative significance).

7.2 Random forest

Random forest consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction. The fundamental concept behind random forest, and the reason that the random forest model works so well is: A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models. The low correlation between models is the key. The reason for this wonderful effect is that the trees protect each other from their individual errors. So the prerequisites for random forest to perform well are: 1. There needs to be some actual signal in our features so that models built using those features do better than random guessing. 2. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

7.3 Support Vector Machine

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence. Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane.

7.4 Naive Bayes

Naive Bayes is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. The algorithm is based on 3 steps:

- Step 1: Convert the data set into a frequency table
- Step 2: Create Likelihood table by finding the probabilities
- Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

8 References

- *Deep Learning-based document modeling personality detection from text*
  Navonil Majumder Instituto Politécnico Nacional, Soujanya Poria, Nanyang Technological University, Alexander Gelbukh, Instituto Politécnico Nacional, Erik Cambria, Nanyang Technological University
- *An Introduction to Statistical Learning: With Applications in R)*
  Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani.
- *Deep Learning*
  Ian Goodfellow and Yoshua Bengio and Aaron Courville