Natural Computing - Project proposal

Alejandro González Rogel, Victor García Cazorla, Tim Bergman

May 10, 2017

1 Problem

Our goal is to build a system that is able to generate film synopsys given some predefined genres. The idea is to have a model that, once trained, will take a set of film topics like e.g. {drama, second world war, thriller} as input and generate as output new film descriptions that fit the given tags.

To train such a model, this project will make use of a film metadata database[1]. This data has been obtained by webpage scraping of the previously mentioned webpage. The SQL file has $\sim 128K$ films infosheets with different fields. Among all the available features (country, director, cast, score etc.), we are only interested in the synopsis and the genre(s) of each entry. One important fact is that all the synopsis and genres are written in the Spanish language, although a different dataset might be considered in the future. A film infosheet example is shown in figure 1.

Figure 1: Film infosheet example of the *filmaffinity* database where the genre and synopsis fields have been highlighted

In this film dataset there are 378 different genres. After filtering out all those instances with synopsis shorter than 5 words or no labelled genre, 113957 instances remain. This is the dataset that we will be working with.

2 Methods overview

2.1 Recurrent Neural Networks

In order to solve this problem, we will implement a Recurrent Neural Network (RNN). A representation of the basic architecture of the network can be seen in figure 2. As an input, we will use a one-hot-encoding vector whose size will match the number different film genres in our dataset. Each position in this vector will represent one of these genres and values 0 or 1 will indicate whether the given instance belongs to that category or not. The expected output will be a synopsis generated word by word.

[1] `filmaffinity.com`

The basic units of our neural network will be Long Short-Term Memory (LSTM) cells [1]. This special type of neuron maintains information about past steps, which allow us to create the recurrent structure. Other variants of LSTM neurons, such as Gated Recurrent Units (GRU) may also be considered if there is enough time to do so. The depth of our network will greatly depend on the amount of computational power that the training will require, i.e we could trade off performance for speed.

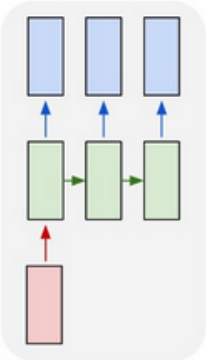

Figure 2: Recurrent Neural Network representation[2]: Each rectangle is a vector of values and each arrow represents one or more layers of a network. The red square represents the input, the green ones the inner state of the network and the blue ones the output.

This way of approaching a text generation problem has been applied successfully in the past. One of the classical examples is image captioning, where the goal is to generate a sequence of words from one single image [3]. The main difference is that in that example the input is an image encoded as a matrix and not a set of labels encoded as vector of zeros and ones.

2.2 Text preprocessing

The synopses do not require extensive preprocessing before they can be used in our problem. Extremely short texts will not be considered and special characters other than letters and basic punctuation marks will be eliminated. We do not consider necessary to perform word stemming, although this decision might change depending on the performance of the network.

The dataset is composed by more than 110.000 instances and film synopses usually contain several sentences. Thus, we expect it to be big enough to train the network without having to deal with a strong overfitting problem.

2.3 GloVe

We will use GloVe, a unsupervised learning algorithm for obtaining vector representations for words that captures its semantics [4]. To achieve this the model first needs to be trained on a big corpus, such as the entire Wikipedia in English (although we will download one that has been already trained and use it to embed the words of our synopsys). At the beginning of the training process, the model efficiently builds a matrix wherein both rows and columns are all possible words of the corpus (or the n most common) and the values are the co-occurrence of each pair. In the following step it reduces the dimensions of such big matrix resulting in a much more manageable one where each word in the corpus is represented by a vector of a fixed size (usually 70-100 dimensions).

2.4 BeamSearch algorithm

Beam search is a heuristic search algorithm that explores a graph by expanding the most promising node in a limited set. Beam search is an optimization of best-first search that reduces its memory requirements [5]. We will use the algorithm at prediction time to sample different sequences of words from the probability distribution that our Recurrent Neural Network will learn.

3 Methods justification

3.1 Recurrent Neural Networks

This project will focus on the text generation problem. Because a sentence (or a set of sentences) is composed by elements that are closely related between them, the use of a RNN helps us to better approach this problem by capturing the sequential nature of our desired output. Moreover, we will use LSTM cells that will allow us to retain some information about past runs and, this way, influence the current output. This will hopefully provide us an increase in performance.

3.2 GloVe

Using a pretrained GloVe model will save us time and resources. For our project we will use vectors trained on the whole English Wikipedia corpus, since it contains pieces of text that resemble the film descriptions of our dataset (in comparison, for example, to vectors trained on movie dialogs). Our current dataset and the fact that the training of GloVe vectors will require extra work and excessive computing power, motivated our decision of using already existing embedings.

3.3 BeamSearch algorithm

The way recurrent neural networks sample the elements of the predictive distribution at each time step is greedy by default, limiting the capability of our model of generate innovative and different synopsys. We do not want our RNN to output the same text when given the same set of genres. One desired feature of our system is, indeed, to generate **different** synopsis given the same set of genres. One plausible solution is to use the beam algorithm to generate variable results, as it is done with Neural Machine Translation with Sequence-to-sequence models [5]. If we apply the idea that is explained there to our problem, it will go as follows: first we predict the distribution of the first word of our synopsis according to the input set of genres; and then we take b samples instead of only one, and so on. This gives quite some variability to the possible output, but the space of hypothesis grows exponentially. Beam Search solves this problem by pruning, and the way it does this is by keeping only the b most likelihood candidate sequences.

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[2] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks, 2015.

[3] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. *IEEE TRANSACTION ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 2016.

[4] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[5] CMU. Neural machine translation. In *Sequence to Sequence Models: a tutorial*, pages 41–43, 2017.