nodes for the scored item efficiently, QS processes the nodes of all the trees feature by feature. Specifically, for each feature *x[i]*, QS builds the list of all the nodes of the ensemble where *x[i]* is tested, and sorts this list in ascending order of the associated threshold γk. During the scoring process for feature *x[i]*, as soon as the first test in the list evaluating to true is encountered, i.e., $x[i] \leq \gamma k$, the subsequent tests also evaluate to true, and their evaluation can be safely skipped and the next feature *x[i+1]* considered.

This organisation allows QS to actually visit a consistently lower number of nodes than in traditional methods, which recursively visit the small and unbalanced trees of the ensemble from the root to the exit leaf. In addition, QS exploits only linear arrays to store the tree ensemble and mostly performs cache-friendly access patterns to these data structures.

Considering that in most application scenarios the same tree-based model is applied to a multitude of items, we recently introduced further optimisations in QS. In particular, we introduced vQS [3], a parallelised version of QS that exploits the SIMD capabilities of mainstream CPUs to score multiple items in parallel. Streaming SIMD Extensions (SSE) and Advanced Vector Extensions (AVX) are sets of instructions exploiting wide registers of 128 and 256 bits that allow parallel operations to be performed on simple data types. Using SSE and AVX, vQS can process up to eight items in parallel, resulting in a further performance improvement up to a factor of 2.4x over QS. In the same line of research we are finalising the porting of QS to GPUs, which, preliminary tests indicate, allows impressive speedups to be achieved.

More information on QS and vQS can be found in [2] and [3].

References:
[1] G. Capannini, et al.: "Quality versus efficiency in document scoring with learning-to-rank models", Information Processing & Management, Elsevier, 2016, http://dx.doi.org/10.1016/j.ipm.2016.05.004.
[2] C. Lucchese et al.: "QuickScorer: A Fast Algorithm to Rank Documents with Additive Ensembles of Regression Trees", ACM SIGIR 2015: 73-82 [best paper award].
[3] Cl. Lucchese, et al.: "Exploiting CPU SIMD Extensions to Speed-up Document Scoring with Tree Ensembles", ACM SIGIR 2016: 833-836.

Please contact:
Raffaele Perego
ISTI-CNR, Pisa, Italy
+39 (0)50 3152993
raffaele.perego@isti.cnr.it

Optimising Deep Learning for Infinite Applications in Text Analytics

by Mark Cieliebak (Zurich University of Applied Sciences)

Deep Neural Networks (DNN) can achieve excellent results in text analytics tasks such as sentiment analysis, topic detection and entity extraction. In many cases they even come close to human performance. To achieve this, however, they are highly-optimised for one specific task, and a huge amount of human effort is usually needed to design a DNN for a new task. With DeepText, we will develop a software pipeline that can solve arbitrary text analytics tasks with DNNs with minimal human input.

Assume you want to build a software for automatic sentiment analysis: given a text such as a Twitter message, the tool should decide whether the text is positive, negative, or neutral. Until recently, typical solutions used a feature-based approach with classical machine learning algorithms (e.g., SVMs). Typical features were number of positive/negative words, n-grams, text length, negation words, part-of-speech tags etc. Over the last two decades a huge amount of research has been invested in designing and optimising these features, and new features had to be developed for each new task.

With the advent of deep learning, the situation has changed: now the computer is able to learn relevant features from the texts by itself, given enough training data. Solving a task like sentiment analysis now requires three major steps: define the architecture of the deep neural network; aggregate enough training data (labelled and unlabelled); and train and optimise the parameters of the network.

For instance, Figure 1 shows the architecture of a system that won Task 4 of SemEval 2016, an international competition for sentiment analysis on Twitter [1]. This system uses a combination of established techniques in deep learning: word embedding and convolutional neural networks. Its success is primarily based on three factors: a proper architecture, a huge amount of training data (literally billions of tweets), and a huge amount of computational power to optimise its parameters. Live demos of various deep learning technologies are available at [2].

Goal of DeepText

In DeepText, we will automate the three steps above as far as possible. The ultimate goal is a software pipeline that works as follows (see Figure 2):

1. The user uploads his or her training data in a standard format. The data can consist of unlabelled texts (for pre-training) and labelled texts, and the labels implicitly define the task to solve.
2. The system defines several DNNs to solve the task. Here, different fundamental architectures will be used, such as convolutional or recurrent neural networks.
3. The system then trains these DNNs and optimises their parameters.

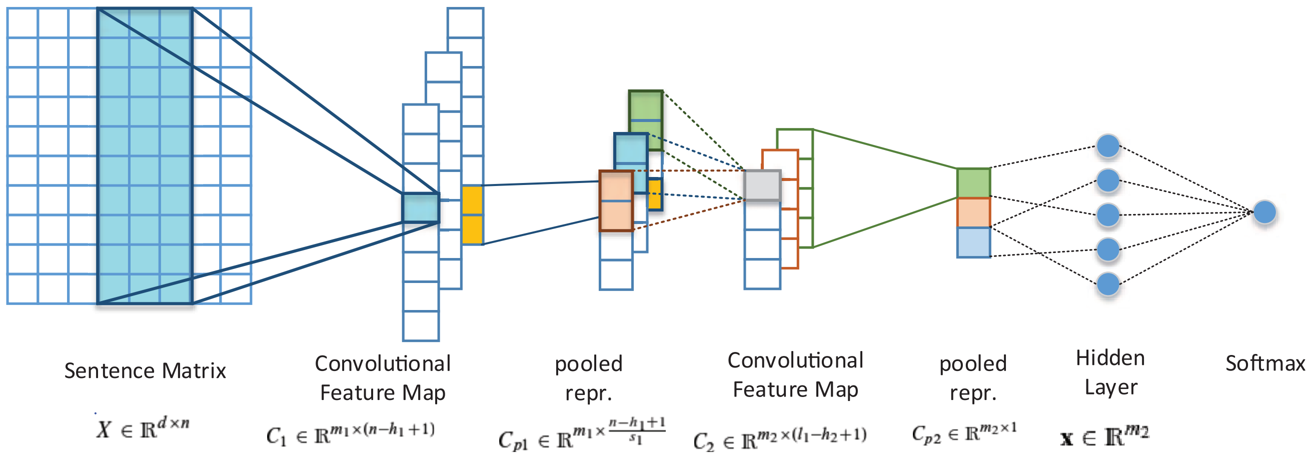
Sentence Matrix
$X \in \mathbb{R}^{d\times n}$
Convolutional Feature Map
$C_1 \in \mathbb{R}^{m_1\times(n-h_1+1)}$
pooled repr.
$C_{p1} \in \mathbb{R}^{m_1\times\frac{n-h_1+1}{s_1}}$
Convolutional Feature Map
$C_2 \in \mathbb{R}^{m_2\times(l_1-h_2+1)}$
pooled repr.
$C_{p2} \in \mathbb{R}^{m_2\times 1}$
Hidden Layer
$\mathbf{x} \in \mathbb{R}^{m_2}$
Softmax

Figure 1: Deep nNeural nNetwork for sSentiment aAnalysis [1].

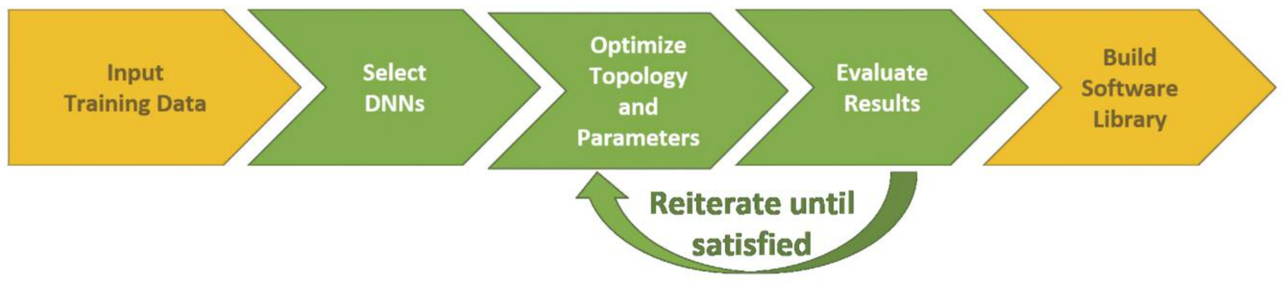

Figure 2: Generating a software library for arbitrary text understanding tasks.

4. Performance of each DNN is measured, e.g. in terms of F1-score, and the best DNN is selected.
5. Finally, the system wraps the “winning” DNN into a software library with a simple interface. This library is ready-to-use in production.

In principle, only the first step – collecting and labelling the training data – needs to be done by humans, since this step defines which task should be solved, and how. For instance, for sentiment analysis on Twitter, each text is labelled with positive, negative, or neutral; on the other hand, if we want to detect companies or persons in text (“entity recognition”), then the proper position of each occurrence of an entity within the text needs to be labelled.

The last three steps in the process above are straightforward, and basically require substantial computational resources and appropriate skills in software engineering.

Challenge: Find a good DNN architecture

The most challenging part is Step 2: to come up with “appropriate” DNNs for the task at hand. There exist several established DNN architectures for text analytics, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs). For each architecture, there exist various parameters: in the case of CNNs, this is the number of convolutional layers, size and number of filters, number and type of pooling layers, ordering of the layers etc. In theory, each configuration of a DNN could be used, but this would lead to an explosion of DNNs to evaluate.

For this reason, we will develop several template DNNs for different types of text analytics tasks: classification, topic detection, information extraction etc. Based on these templates, the system will run a pre-training where each template is applied to the task at hand and evaluated. Only the most promising DNNs will then be used for parameter tuning and optimisation.

Our goal is that, given the training data, the system will generate a suitable software library within three days.

About the Project

Deep Text is an applied research project of Zurich University of Applied Sciences (ZHAW) and SpinningBytes AG, a Swiss startup for data analytics. It started in 2016 and is funded by the Commission for Technology and Innovation (CTI) in Switzerland (No. 18832.1 PFES-ES).

Link:
[L1] http://spinningbytes.com/demos/

Reference:
[1] J. Deriu et al.: “SwissCheese at SemEval 2016 Task 4”, SemEval (2016).

Please contact:
Mark Cieliebak
School of Engineering, Zurich University of Applied Sciences (ZHAW) , Switzerland
+41 58 934 72 39
ciel@zhaw.ch