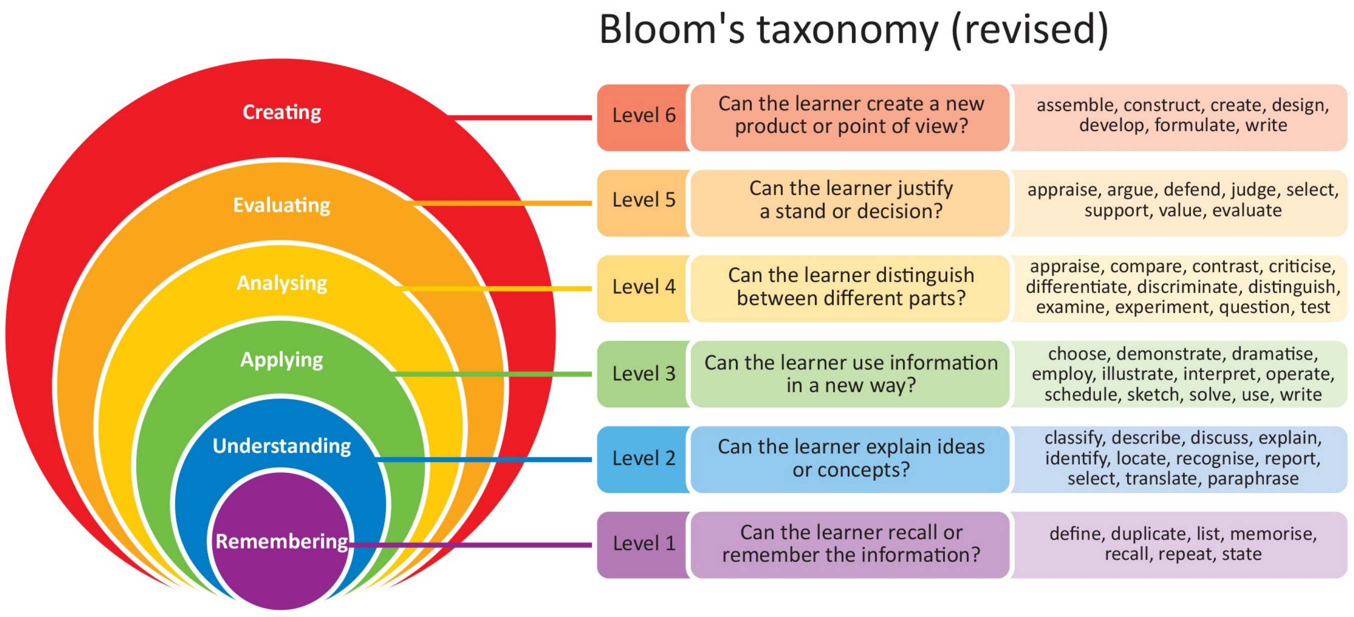
Recurrent Neural Networks for Bloom's Taxonomy Classifier

This post summarizes an initial one-week prototype cycle of building a machine learning based text classifier to identify the context, according to Bloom’s digital taxonomy, in a given question. The technologies used are recurrent neural networks with LSTM cell units from tensorflow.

Bloom’s Digital Taxonomy

Bloom’s digital taxonomy system is a tool for teachers and instructional designers. The system classifies a learning object (e.g. course, lesson or question) according to a scale that reflects increasing levels of cognitive skill. The scale ranges from “remembering“, which is the lowest level of cognition, to “creating“ as illustrated in the picture below. For example, a given lesson that teaches you “How to Identify the differences between two species.“ would fall under the “understand“ category according to Bloom’s digital taxonomy; While on the other hand a course named “How to build an integrated circuit amplifier”, would probably be associated with either “applying“ or “creating“ or both. For more details on Bloom’s taxonomy, you are encouraged to refer to specialized articles, such as [1,2].



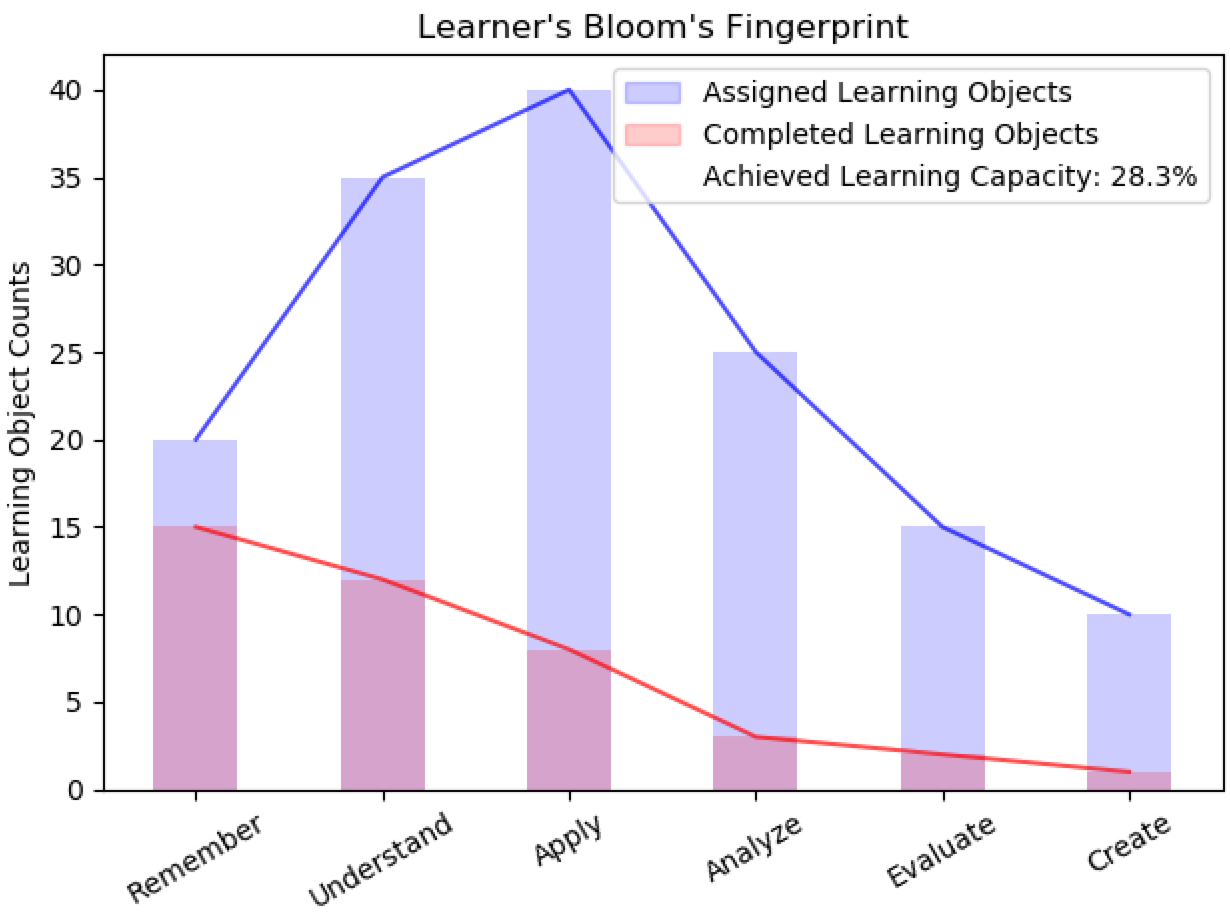
Level 6	Can the learner create a new product or point of view?	assemble, construct, create, design, develop, formulate, write
Level 5	Can the learner justify a stand or decision?	appraise, argue, defend, judge, select, support, value, evaluate
Level 4	Can the learner distinguish between different parts?	appraise, compare, contrast, criticise, differentiate, discriminate, distinguish, examine, experiment, question, test
Level 3	Can the learner use information in a new way?	choose, demonstrate, dramatise, employ, illustrate, interpret, operate, schedule, sketch, solve, use, write
Level 2	Can the learner explain ideas or concepts?	classify, describe, discuss, explain, identify, locate, recognise, report, select, translate, paraphrase
Level 1	Can the learner recall or remember the information?	define, duplicate, list, memorise, recall, repeat, state

Image from Niall McNulty

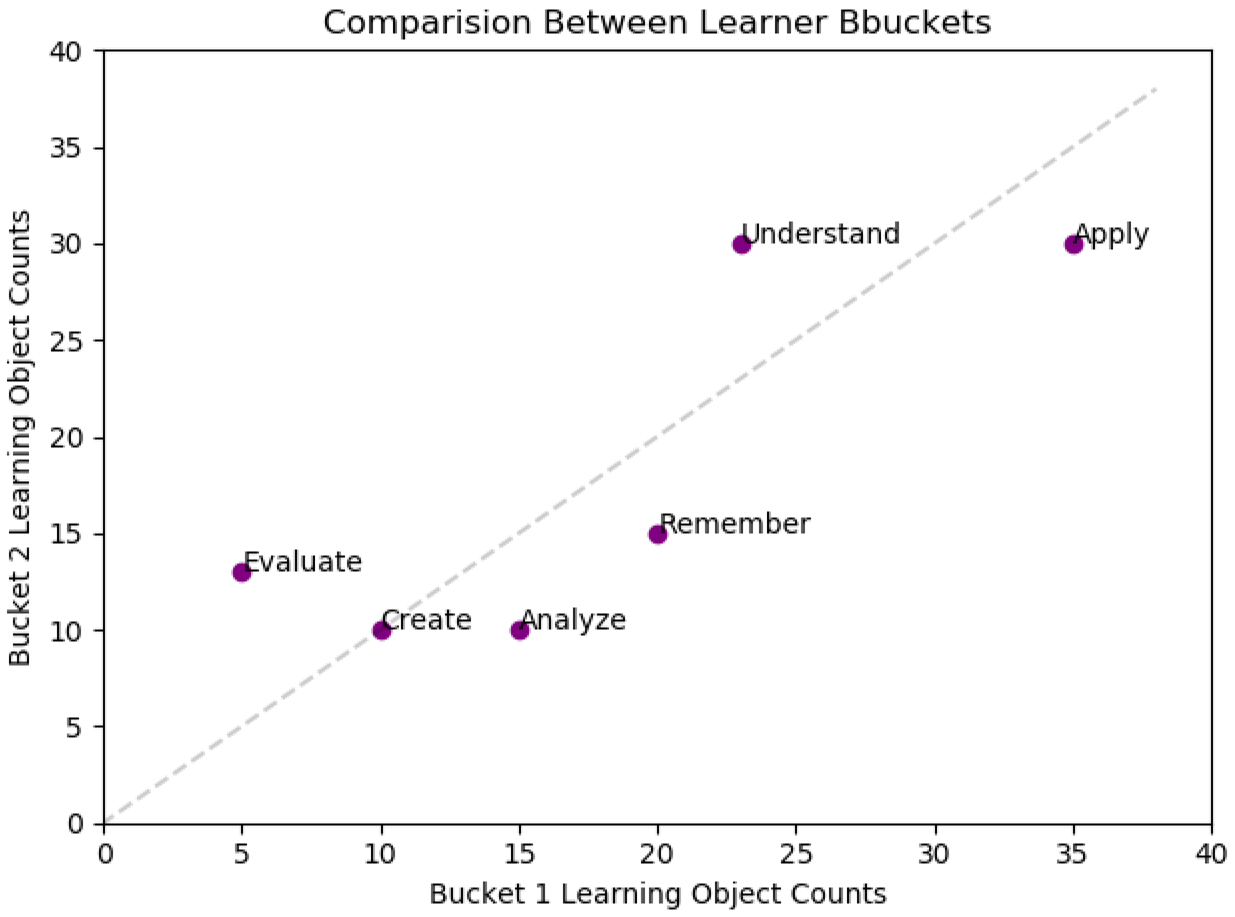
Learning object taxonomy in e-learning

In the e-Learning domain implementing a taxonomy system is a fundamental feature. In addition to driving course design; It also carries valuable metadata information regarding the learning objects themselves, which can be further utilized. For example, a learning content oriented taxonomy system, such as Bloom’s, could potentially enable advanced learning objects ranking as well as learner segmentation based on a cognitive scale. Subsequently, this type of ranking could be used to drive recommendation engines as well as adaptive learning strategies, that aim at progressing learners through certain areas in the cognitive level space.

A potential way to visualize learner progress in the cognitive space is illustrated in the following bar chart; Where all the accumulated questions that the learner followed around a given topic can be summarised.



In the previous figure, the vertical axis represents the number of accumulated learning objects. The percentage of learning objects completed with respect to the assigned ones could be called, “Achieved Learning Capacity“. A similar bar chart could be built around the learning objects that a course or lesson includes during course creation. Thus, offering the opportunity to visually inspect the cognitive levels covered by the course or lesson. Having such metrics might be useful to identify gaps during course design or during a learners progress around a topic; Or even extract similarity metrics between courses or learners based on their accumulated cognitive skill content or profile respectively.



In the last figure, a simple comparison of the accumulated learning objects between two learner buckets is shown. The two buckets have equally progressed in any of the six different cognitive level indices if the respective index is on the diagonal.

LSTM cells for Bloom's classifier

It is quite common in e-Learning to implement Bloom's taxonomy and it is not a new idea to build a classifier to identify the bloom's index given a question [3,4,5,6]. The motivation behind this post is to share the experience while prototyping a Bloom's taxonomy classifier using state of the art deep learning and natural language processing (nlp) techniques, specifically, recurrent neural networks (rnn) with LSTM cell units. The excited user will probably be disappointed as it is well-known that with deep learning and nlp one typically needs a lot of data to train the huge number of parameters; And unfortunately, in our case, we had a minute dataset of 600 questions. Thus, no robust claims about comparing classifiers can be drawn from this post at this stage of analysis. However, there are a few lessons to take upon in the next analysis iteration, where we are looking at a much larger dataset. But lets first see a few key points of the technologies involved before passing to the conclusion.

Recurrent neural networks alongside with LSTM cells are typically used for sentiment analysis with great success since they offer a way to model sequential, time-series like, data much easier and efficiently while capturing semantic relationships in the text. The mathematics behind is quite interesting, see [7,8,9], and there is a great software support for most deep learning frameworks, like tesorflow, dl4j, e.t.c.

Furthermore, machines cannot process text directly, thus a big issue when dealing with text has to do with the numeric representation of words as vectors, also called word embeddings. There are different ways to do that, such as the commonly used tf-idf. However, most of the standard techniques fail to capture the semantic relationship between words in a sentence, which is crucial in sentiment-like classifications of text like the one we are building. Thus, two approaches were followed in the current short study.

First, we used pre-trained embeddings from the Stanford university glove and google-news-vectors language models. These embeddings were trained based on a huge corpus of texts, where essentially a deep learning neural network system, like gensim, is learning how to predict a missing word in a given sentence. In the second scenario, we obtained the word embeddings as part of the rnn training phase, which is quite straightforward in tensorflow. The second approach has more parameters, implying slower convergence and more prone to overfitting. However, it seems that more semantic context in terms of Bloom's taxonomy is captured in these word embeddings compared to the pre-trained ones. This is probably because the embeddings vary just like the rnn weights and biases do during the training phase and are thus optimized in the same loss function. Implying that the word embeddings are optimised more for better discrimination between the different bloom's taxonomy categories.

Conclusions

After a week of prototyping and “pseudo-tunning” we reached an accuracy of about 73 percent and an average f1 score of about 0.5. We used a labeled dataset found online that was used for building a similar classifier with support vectors machines, which reached an accuracy of about 80%. Our trained model, most likely, overfits the data thus, you should not go and implement a fully tuned rnn based bloom's classifier without prototyping with more data first, and testing with several testing samples; Given this result we are compelled to consider the initial 5000 learning outcomes sentences, that we first need to manually label, and the more than 60k questions related to these learning outcomes.

Lastly, regarding the word embeddings; As mentioned above during the training phase of the rnn the word embeddings can be also fitted to capture more context related to Bloom’s taxonomy. What is typically done with the word embeddings, is to compute the cosine distance between words and see how similar two words are, (parallel-similar vectors are at cosine distance of zero). Stimulated by this idea we summed, as vectors, the word embeddings inside a given question and extract the corresponding “question embedding“. Our hope was to plot the question embeddings in a reduced space, using pca, and see that questions cluster in six distinct Bloom’s categories. However, as expected due to the limited data sample, the information loss of pca as well as the large number of, six, categories produced a random looking plot. However, we at least wanted to observe a hint that the trained embeddings are indeed capturing some context. So, we computed the average separation (based on cosine distance) between the above-mentioned question embeddings vector clusters in two scenarios and found something interesting. In the first scenario the cosine distance between question cluster vectors is computed for the best performant model; While in the second, the labels were on purposely shuffled such that they wrongly label a question and thus the embeddings capture random context from the questions. We were happy to observe that the distance separation increases in the first scenario by almost 3 standard deviations; Implying that the embeddings indeed capture some of the context of questions. It is remarkable to realize that an algorithm can probe information about blooms taxonomy from the semantics of the language without explicitly being externally constrained to any rule about the language.