SemanticSky

Suppose we have a set of text-based documents and we want to rank evaluate all 2-place similarity relations occurring between them. Following a typical machine learning approach, we would then first extract a list of features from each document. Secondly, we would write a few algorithms that compare such features pairwise to induce equivalence classes of documents (for similarity is an equivalence relation). At that point we are likely to discover that different algorithms have different strengths and weaknesses, which we might think of as expertises. Suppose for example that (either because the data is partially annotated, or because we do some smart preprocessing beforehand) we know, for each document, the author, title, the content and a list of tags. Suppose also that there are big differences between the quantity and quality of the content of each document: some documents have many tags, some have none, some documents lack an 'author field', some have four authors. Then, the algorithms that base their evaluations on these features will clearly give results that are influenced by the quality of the data. But, taking this step further, we can also imagine that our documents belong to different classes, or groups, such that each class has some typical features that other classes totally miss. We can see this as identifying areas of expertise for the algorithms.

Once we have noticed that each algorithm is better than others in a subset of the input spaces, we will naturally ask ourselves how we can combine their inputs in a smart way, such that each area of the input space is covered and that we exploit the areas of expertise of each algorithm in the best way. Ideally, we would like having some sort of round table where the experts contribute each according to its strengths, and stay silent where they have nothing to say or where they know they are not keen.

This is the situation that SemanticSky tries to achieve, by employing a supervisor algorithm that gates and collects the various algorithms' opinions and merges them into a single one, trying to make best use of their individual capacities.

Ideally, we would like having some sort of round table where the experts contribute each according to its strengths, and stay silent where they have nothing to say or where they know they are not keen. Making consistently good decisions on pairs of documents of a particular content-type would then result in a high contextual self-confidence for that category of documents, and vice versa. Thus areas of expertise arise for each algorithm through a feedback system that affects the algorithms' self-confidence relative to the content-type of what they are evaluating upon. Such self-confidence is then

used to weight the raw algorithmic output, and the suggestion thus obtained is forwarded to the supervisor, that updates its belief state therewith.

The system as it is now relies on partially annotated data: the content-type is handmade into the documents' metadata and the documents come with the author, headline and other useful information.

Running a few tests we found out that the system's performances were heavily negatively influenced by two main biases: an *output bias* that consisted in most algorithms outputting decisions numerically on average too low; the normalization of the output in the [0,1] space produced an uneven distribution of values towards 0. The second bias was a *sample* problem: there were disproportionately many negative examples and too few positive ones for the algorithms to learn on: this resulted in contextual trustworthiness rapidly tend towards 0, following a 'it's statistically most convenient to always say 0' strategy.

Early attempts to counter these issues resulted in a differentiation of learning rates attempt and an equalization system. The first one is rather self-explanatory: we reduced learning rates for negative examples by a factor of 50; the second one is a bit more complicated. First we individuated the average value of suggestions that were proven to be correct by feedback, and the average of the incorrect ones. Then, we picked the middle point and used it as the centre of a repulsion force to separate the good suggestions' values from the bad ones'.

At a higher level, one can think at SemanticSky as a multiagent system where agents try to maximise good feedback (i.e. minimise their regrets) by adjusting their self-confidence ratings about the opinions they have (which can interchangeably be seen as the supervisor's trust in their opinions).

The supervisor algorithm simply represents the top-level belief set, or learner, and can also act as a gating machine (if only we gave him some sort of equalization) to adjust the suggestions of the agents even beyond their trustworthiness' limit.

By writing appropriate algorithms, the space of possible inputs for a SemanticSky implementation can be expanded in various directions: by extracting appropriate features and having algorithms capable of crunching them, a SemanticSky could evaluate 'longer-than', 'more-beautiful-than', 'as-likely-to-be-interesting-to-$Johnny-as' or virtually any other kind of relation on musical objects, books or faces of people. Before we reach that stage, however, the framework will probably need some more refinement to counter the problems we already pinpointed, and these we will eventually discover in the future.

Pietro Pasotti, August 2014, Amsterdam.