Commodity AI?

Follow

Published on 2020 M01 192020 M01 19 • 1 Likes • 0 Comments

Simon Laub

**Cand.Scient. Mathematics & Computer Science. Thesis in AI.
Interests: Cognition, natural- and artificial intelligence.**

According to the dictionary, a "commodity" is a an unprocessed or partially processeed good. Grain, fruits, metals are commodities, a service is not.

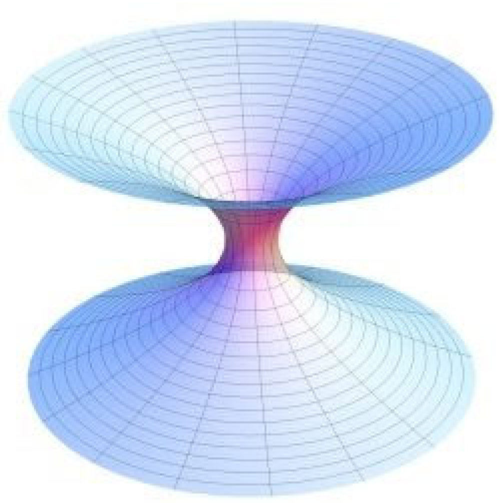
In the IT world it has recently been argued that AI is rapidly becoming a kind of "commodity" (1). Included in a product, as a semi-finished product that another service just happens to use. And why not? IT programs are rarely monoliths anymore. Most programs include a lot of third party software, where programmers only know how to use these sub-programs, not how they actually work. And the big players in the IT world (IBM, Google etc.) certainly appear eager to make AI components available to everyone. Kevin Kelly (Wired Magazine) has drawn the obvious conclusion: "The business plans of the next 10,000 startups are easy to forecast: Take X and add AI"(2).

But how easy is it really these days to "Take X and add AI"? Can just about anyone do it? Well, as usual, the devil is in the details. What is "X", and what do you mean by AI? What kind of skills do you have? How much time do you have?

Sure, in a real project, you might have an idea about how much time you yourself would need in order to "add AI to X". Or you could ask your team, whether anyone would be interested in the task, and how much time they think they should use. Or you could hire in consultants with the exact expertise that you need. All fine, but what do you do, if your boss asks you to "Take X and add AI" in an IT project, where no one has an AI background, and will have to make do with on-the-job-training? Do you think such a team could do the job in a couple of months? A year?

Well, about a year and half ago, I found myself in that exact situation, and could hear myself saying something along the lines of "it all depends"... Not exactly a brave answer, and also a bit unsatisfactory. Surely, it should be possible to do better?

Time for some experiments!



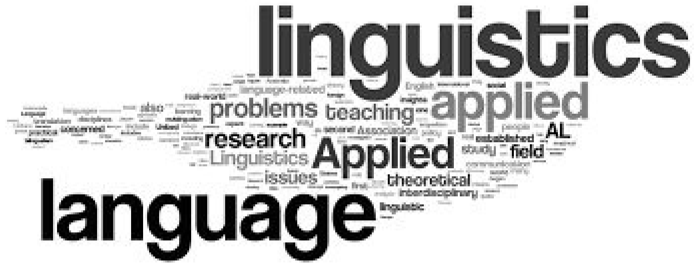
Working in a educational environment it is not that difficult to find (or come up with) projects (our X's), that could be improved with a little AI. And it is also relatively easy to find students who (as part of an internship or an assignment) will like to be part of a team, and do most of the actual work. Sure, not exactly an "Take X and add AI" project in a small private company, but close enough to be interesting. Could this be a success?

For our "X", I decided to take a look at classification of job adverts into job categories. A lot of people around me were talking about it at the time. It sounded challenging, and with a nice "ai-feel" to it. I.e. our institution, and similar institutions in Denmark, are very interested in knowing what kind of skills employers are looking for. As it is basicly those skills that we are all supposed to give our students. So, it would be nice to know exactly what kind of skills the employers say e.g. a Java programmer should have.

Long before we started on this "Take X and add AI" project, our institution, in cooperation with similar institutions in Denmark, has in recent years been scraping the internet for job ads. They have also been searching for words in the ads, using regular expressions, in order to classify them. An ok approach, but this relatively straight forward process wasn't able to classify all of the ads 100 % correctly. Some job ads weren't using the exact keywords the program was looking for. Indeed, natural language processing isn't that easy. And the project had already called in AI consultants to see if they could help with some of these more tricky problems they had encountered.

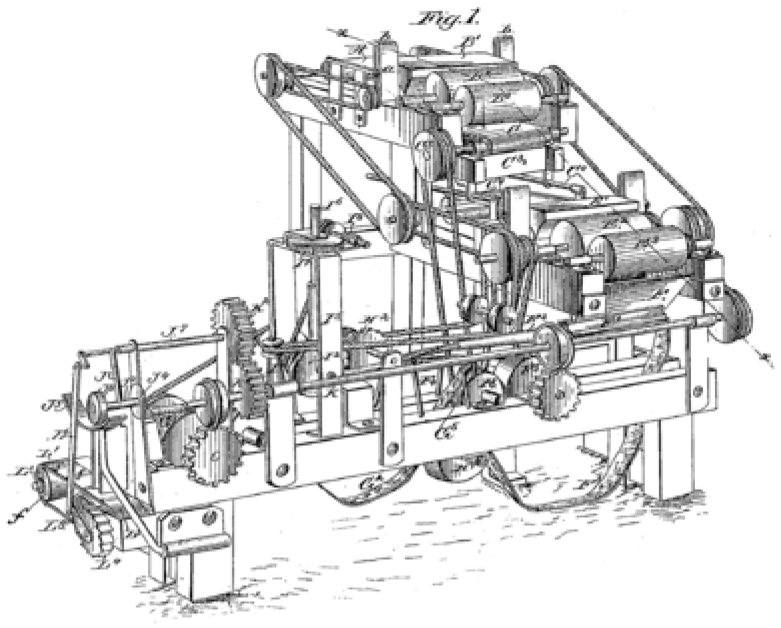
Our student teams would work in parallel with these efforts and take a closer look at some of the same "job ad" classification problems. So far, so good, it certainly sounded like a "Take X and add AI" project, where the AI component would be some kind of text processing component. Indeed, we were in good company, as text processing is the core business of todays major internet companies, companies like Facebook, Google, Twitter, Baidu, Yahoo etc. Starting in 2018, 4 students were given 10 week internships to help us with this AI natural language classification problem. Almost finished with their 2 ½ year AP degree in computer science they all had a solid background in IT, but no experience with Machine Learning or Artificial Intelligence.

The 2 first students decided to work together, and quickly became convinced that the way forward was [Tensorflow]() and [neural nets](). As they had no prior experience with neural nets their project was focussed on getting to know Tensorflow, and making the "job ad" data available for a neural net. Preparing, and getting to know, the data is, of course, a major part of most data science projects, normally consuming as much as 80 % of a projects time and efforts. But it quickly became clear that we were trying too many different things in this first project. The students were not only asked to come up with a good way forward to "Take X and add AI", they were also asked to make an actual working prototype for doing this. Given the time-constraints the students ended up deciding to focus on building a TensorFlow prototype. Without thinking too much about the many other possible ways forward.



But how, exactly? In natural language processing, the "Distributional Hypothesis", states that words that are used and occur in the same contexts tend to purport similar meanings. So, it was decided to include a word embedding layer to the pipeline. This "word embedding" layer would then map semantically similar words into nearby points in a vector space. I.e. the system maps 'Obama' close to the word 'president', 'media' close to 'press', 'speaks' near to 'greets' etc., in this abstract vector space, after training on a large text corpus. And adding this preprocessing layer to the pipeline should then make it easier for the neural net afterwards to classify the job ads correctly. Then there were the technical details of setting up the neural net itself in TensorFlow. Tuning hyperparameters for the neural net, things like: a) How many hidden layers should the net have, with how many neurons b) what activation functions should be used, c) What kind of optimizer should be added to the neural nets gradient descent mechanism. Etc.

10 weeks passes quickly when you are having fun. And the first student project was happy to end up with a working prototype based on TensorFlow. A neural net had been trained on jobs ads (labelled by hand into correct categories), and was then able to tell us with 90 % accuracy, whether a new job ad was a "Java programmer" ad or not. So far, so good.



The next 2 students decided to work separately, and were asked to improve on the first student groups work. One student quickly decided that "Adam Optimizers", "Lelu activation" functions and what have you in the TensorFlow universe was a little bit too much to take in, and decided to shift to another intern project. The other student struggled on, but was somewhat overwhelmed by the amount of TensorFlow vocabulary to learn, just to understand what the prototype was doing. Not to mention that he was supposed to improve on the previous groups solution.

To say that there were many ways to move forward would be an understatement. A new neural net could work in an ensemble with the old one, adding its results to the old systems results. I.e. one could train a neural net on new classifications, like "how technical is the job". Where a shop assistants job would be considered not very technical, whereas our Java programmers would be given the classification "very technical". Adding such extra classification results to the other neural nets classification might improve the overall accuracy. Adding the classification results from the old "non-AI" system to the ensemble was another possibility. And then there was Facebooks opensource "FastText" that promises to be able to do text classification. Indeed, all of these results could potentially be added to an ensemble solution. And what about simple mechanisms like just counting words, in a "bag of words model". Maybe that would be helpful?

It was time to look at the machine learning project checklist (See: Appendix B, “Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow”, 2nd Edition. Aurelian Geron.) **1)** Frame the problem and look at the big picture. What is the business objective for minimum performance? **2)** Get the data. **3)** Explore the data to gain insights. Study how you would solve the problem manually. **4)** Prepare the data to better expose the underlying data pattern to Machine Learning algorithms. **5)** Explore many different models.

Clearly, there was more than enough to work on here. Again, in 10 short weeks.

The proper way to proceed? Doing a proper data analysis, and fully expect this to take the usual 80 % of a data science projects time. Probably. But what about results? We all want to present results, working prototypes, insights from a data analysis or similar. Quickly. The student project had reached a rough patch of bad weather.

Thinking a little about it, the situation was probably not that different from something you could experience in just about any small company dealing with a new problem. How can you produce results while you are in the middle of doing research, and learning on the job? After having talked with me about it, the student decided that you can't really expect to produce results in the middle of learning and doing research. Still, the student was very eager to produce results, so it was decided that he should focus on making a small incremental improvement to the existing prototype. Indeed, doing the first 5 steps in the checklist properly, not only needs backing from management, and time, it probably also takes experience to withstand pressure (from yourself) to come up with concrete results right away. A postponement of the glorious moment when everything just works beautifully.

In the end, the second student project ended up with an improved prototype, based on an ensemble of neural nets, this time capable of classifying 4 job groups with a higher accuracy than the first prototype. All very cool indeed.

And adding more tricks to the ensemble, in an incremental way, will then be a natural starting point for coming student projects.

But, commodity AI? Not really. "Taking X and add AI" may sound easy, but is is probably still a lot harder than it sounds, especially if you don't have a lot of ML/AI experience. And yes, you also, still, need a lot of good-old fashioned IT experience. We are not quite there yet there, where it is all super easy...

These days everyone is talking about being agile, and having prototypes up and running quickly. But things like data preparation still accounts for about 80% of the work for most data scientists (3). And an experiment in a Jupyter notebook (favourite tool in many machine learning projects) will often be just as good as other kinds of prototypes, and more than enough for most small experiments. Indeed, skipping parts of the analysis phase was probably never a good idea. Certainly, many studies have told us that it is a bad idea to reduce the costs of the analysis phase (as percent of the total project cost). Below a certain threshold, say 10 % of the total costs, the whole project will suffer, and probably end up being massively over budget... (See e.g. p. 172: Kousholt, B., 2017. "Projektledelse - Teori og praksis". Nyt Teknisk Forlag).

Indeed, project management theory obviously also applies to "Taking X and add AI"- projects. Just as it is still important to be precise and realistic about what you want to achieve. "Learning on the job" is fine, but it takes time. Just as doing experiments takes time, that all should be included in the plans. No matter how eager everyone is for coming up with results.

So, how did we do on the "Take X and add AI. Can just about anyone do it" question ?

Well, the answer is probably still "it all depends". AI/ML experience is good. IT experience is good. As is a realistic project planning. But with these things a lot of new, and very exciting, things certainly becomes possible. Things that weren't possible just a few, short years ago.

Aarhus. January 19th, 2020. Simon Laub.

Simon

Udgivet af

Simon Laub

Simon Laub

Cand.Scient. Mathematics & Computer Science. Thesis in AI. Interests: Cognition, natural- and artificial intelligence.

Følg