Better Glue for Pipelines

504: Machine Learning meets Program Analysis Assignment 1

Luheng He

luheng@cs.washington.edu

Softwares used to solve natural language processing (NLP) and machine learning (ML) tasks are usually pipelined and consist of a series of subtasks. In Table 1, the subtasks in blue color are mostly task-independent, and usually harder to develop. For these tasks, people usually depend on well-developed open-source softwares, such as NLTK, Stanford NLP (for tokenization, tagging, parsing, etc.) and LibSVM, Mallet, Torch (for learning). The other subtasks, such as input reader, feature extraction and evaluation, are highly task-dependent and have to be developed from scratch. Sometimes people are required to use a published evaluation script (in Perl, for many NLP tasks) for the purpose of fair comparison. For NLP tasks in particular, due to their[1] structured nature and huge input/output space, there is really no ideal software that fits the requirement of all tasks.

	Typical subtasks for NLP	**Typical subtasks for ML**
1	Input Reader	Input Reader
2	Segmentation/tokenization	Pre-processing/Data filtering
3	Pos-tagging/Parsing/Named-entity Recognition	
4	Feature Extraction for the target task	
5	Parameter Fitting (Learning)	
6	Evaluation/Cross validation	
7	Hyper-parameter Tuning/Model Ensemble	
8	Output/Analysis/Visualization	

Table 1: Sub-tasks in NLP/ML pipelines.

Therefore, in a typical pipelined NLP/ML software, developers are usually forced to alternate between 1). software written by themselves and by someone else. 2). software written in

Summary of Comments on he.pdf

Page: 1

Number: 1 Author: mernst Subject: Highlight Date: 1/12/2016 6:01:06 AM

I don't see how these aspects are related to the rest of the paragraph. The rest of the paragraph seem to be about the architecture of the system and composing different parts of the software. This is about the structure of the problem. It's the first mention of this in the paragraph yet it seems to be the conclusion and I don't see how it is related to the rest.

different programming languages.[1] Glue code are written to pass around data and parameters. For example, someone would write a some data processing script in Python to scrape all the English titles from Wikipedia, dump them into a file, use Stanford NLP (in Java) to parse the titles and store the results into a list of objects, write some Java code to extract features, learn a classifier using the Java LibSVM wrapper. And for evaluation, what one might do is write the prediction in a text file and run a Perl script from the commandline.

There are many things that can be improved in this pipelined process. The most important one is probably about the glue code written to connect the subtasks. Glue code is usually developed in a hurry, lack developers' attention, and almost never get tested or verified.[2]

Here is a realistic case: When using files to pass around intermediate data, developers need to keep track of the file format. Sometimes Mthe file format is specified in the documentation, sometimes in comments such as /* {word} \t {tag} \t f1, f2 …. */, and sometimes in the developers' brains. Misunderstanding and misalignment in file formats cause errors. For example, the syntax parsing software outputs a parse tree with word indices starting from 1 but I thought they start from 0.

Possible improvements to this [3] problem would include:

1. Write [4] comprehensive comments/documentation for the Input/Output file formats or objects.
2. Write unit tests for the Reader/Writer functions in the software.
3. Write methods to check the validity of input files/objects.
4. Provide sample Input/Output data along with the software.

However, doing one or more of the above suggested could slow down development. It would be great to come up with tools to automatically generate/suggest test cases/sanity checks for the Reader/Writer methods. Maybe we can come up with an easy way for developers to specify the types and the constraints of the intermediate data objects between the subtasks, and automatically generate tests/verifications to check those types and constraints.[5]

Page: 2

Number: 1 Author: mernst Subject: Highlight Date: 1/12/2016 6:03:34 AM

These characteristics are true in many domains. For example, when writing a web application there's usually a framework that provides most of the functionality and the user creates callbacks and other small aspects of the system. Many other examples exist. So I believe that the problem is more general, though I'm glad you are choosing one specific instance of it to focus on, because that will make it easier for you to come up with concrete ideas and more likely to produce something that is practically interesting and capable of being evaluated.

The problem of multilingual software is perhaps a bit less prevalent, but still common.

Number: 2 Author: mernst Subject: Highlight Date: 1/12/2016 6:05:36 AM

This is a discussion about the methodology or attitudes of the developers. Can you be more concrete about the problem that you wish to solve? I don't think you're proposing to change the psychology of these programmers...

Is the problem that this code is developed in a hurry? That is, is this approach effective overall letting people get their research results? Can you argue that it is counterproductive and some other approach would be better overall?

Number: 3 Author: mernst Subject: Highlight Date: 1/12/2016 6:08:29 AM

You haven't stated a problem. You stated a general methodology and only hinted that you think it is very good, but you haven't's set specific things about what is wrong with it. This paragraph actually makes the problem more concrete, but it only mentions for distinct problems at an extremely high level and doesn't give any specifics about the concrete difficulties that programmers face nor how they might be solved. So I feel that in this document you've laid out an application domain and said something about the methodology but you haven't made clear what one or two problems are, why they are problems, how you know what, what programmers do currently, and how you might improve that in the future.

Number: 4 Author: mernst Subject: Highlight Date: 1/12/2016 6:06:55 AM

Can you define this term or give a metric by which documentation can be tested to determine that is comprehensive? Without that, it's unclear exactly what the goal is.

Number: 5 Author: mernst Subject: Highlight Date: 1/12/2016 6:09:13 AM

The suggestions are very general. I could be better to focus on a smaller number of problems and to be much more specific about how they might be solved. Even if your solution ideas are not ultimately successful, this will lead you along the way to finding successful ones.