It's a match! Fuzzy string comparison for interactive learning resources

Capstone Proposal

Oliver Tacke
January 20, 2017

Proposal

It was more difficult than I thought to come up with a proposal for a capstone project. There are tons of interesting datasets out there and dozens of questions that you could possibly ask, but I'd love to create something that someone actually needed. I contacted a friend of mine who co-created OpenSNP, a platform hosting raw genotype data. In fact, he could need someone to use labeled data from 1000Genomes in order to create a classifier that would tell you what's the origin of genetic data. Unfortunately, my knowledge of biology is too poor. Anyway, what I learned form this inquiry is how important domain knowledge and interdisciplinary teams are for data science and machine learning.

I chose to have a closer look at the field of education where I know a thing or two. I found a paper that dealt with undergraduate student generic problem-solving skills. It was based on an empirical study that had produced some data. It could have been used for creating a predictive model for problem solving skills performance, for detecting/classifying sub-groups of students, etc. Unfortunately, the data was not freely available. We need more Open Science! I contacted the author, but I guess he was more afraid of "losing" his data than excited about getting new tools he could use. He wanted to have my resume - that he received - but I never heard of him again.

Eventually, I stumbled upon two nice datasets that might allow to predict the sales figures of video games based on scores from critics. I completed a proposal for a capstone project, but then I decided to at least pursue an idea that I had mentioned briefly. However, it might be out of scale for this capstone project and I should probably go for it later on . . .

Domain Background

In recent years, there have been many initiatives to develop and publish Open Educational Resources (OERs). Those "are any type of educational materials that are in the public domain or introduced with an open license. The nature of these open materials means that anyone can legally and freely copy, use, adapt and re-share them. OERs range from textbooks to curricula, syllabi, lecture notes, assignments, tests, projects, audio, video and animation." (United

Nations, 2016) Living up to this attitude, in 2012 the Norwegian organization *National Digital Learning Arena* funded the initial development of H5P. H5P is an open-source framework that allows to create, share and reuse interactive content, e.g. interactive videos or presentations with different kinds of quizzes.

For the last six month, I have been involved in the development of H5P in my spare time. Just recently, I created a new feature for clozes that is currently pending as a pull request on github. With my proposal, users could create clozes that do not necessarily require an exact answer. For example, if in a chemistry quiz someone typed "deoxiribonucleid acid" instead of "deoxyribonucleid acid", he or she probably meant the right thing. A teacher might decide that this minor wrong spelling should not lead to the answer being wrong. The same goes for names such as "Schrödinger", "Schroedinger" or "Schrodinger". While you could also give alternative spellings explicitly, this might be tedious in some cases. For instance, transcribing Russian names or anticipating typical spelling errors might be tricky.

While I basically implemented a nice feature that can already be tested on my blog, it has a downside as discussed on the H5P forum. On the one hand, users can fine-tune the related algorithms to their specific needs using some parameters. On the other hand, it is not obvious for unexperienced users what the parameters will do. It might be easier for them if they could just switch "fuzzy comparison" on or off. This would require the algorithms to be tuned at their best. That's what I intend to do with this project.

Problem Statement

So far, I implemented two common algorithms (or four with variants) that will take two strings as input and return a numerical string distance. The string distance measures the similarity of both strings. Users can define a threshold for each function that shall separate wrong answers (low similarity) from correct answers (high similarity). In consequence, I am facing a classification problem. My goal is to find the "best" classifier possible without the need to be fine-tuned by the user.

Datasets and Inputs

The biggest challenge for this proposal is probably the availability of a suitable dataset. I don't think there is one. I'll have to create one on my own. Possible columns could be:

- **master word:** The correct solution to compare with (string)
- **variation:** master word with character insertions, deletions, substitutions or swappings (string)
- **difference:** number of character insertions, deletions, substitutions or swappings (integer)

- **match:** human rating of the variation (boolean)
- **complexity/commonness:** human rating (integer [1..3])
- **length:** number of characters of the master word (integer)
- **language:** language code (string)

This dataset would consist of examples that contain a master word and a similar string. Similarity would also be stored as the number of character insertions, deletions, substitutions or swappings used on the master word to create the variation. This number basically is the Damerau-Levinshtein distance. The most important thing to store would be the human ratings of similarity - either it's a match or not. Those are my labels. Of course, each person should have to rate different variations of a master word. Since different people can have different judgements about similarity, I also want to have multiple examples for the same master word/variation combination with labels given by different people.

My ideal dataset would also contain an indicator of complexity or commonness, because errors for complex or uncommon words might not be as severe as for simple or frequently used words. This feature should also be added by different people. Furthermore, some additional features might be generated or obtained automatically, e.g. string length (some metrics may be better suited for certain word lenghts) or language (some metrics may work better for certain languages). And, of course I want to generate the results from several algorithms later on.

Data might be collected using a crowd sourcing approach. A website could show people a word and ask for an estimate of complexity or commonness. Afterwards, the website could present 10 variations of varying similarity and ask whether the person would rate them as a match or not. This is rather boring, but maybe Gamification or some other form of motivation might help. And why not build a Tinder-esque GUI . . .

For me, it's hard to anticipate how many examples would be necessary. Using a rule of thumb approach to generate an artificial dataset and plotting the learning curves (training error and the cross validation error in relation to the training set size) might give me a hint. I guess, my minimum goal would be to collect data for 1,000 words with 10 variations each and ratings from at least 5 different persons (per language). Consequently, 5,000 people would have to rate 10 word pairs.

Also, I might hava a look at the APIs that Google is offering for their search engine. I could present a misspelled word to the search engine and compare it to Googles suggestion. If both are identical, then we have a match (thanks to Anja Lorenz for this idea).

Solution Statement

The solution will be a classifier. It will take two strings as input values and either classify them as similar enough to be a match or not. There are several options. The simplest one would be to optimize the separation threshold for each algorithm individually based on just the two strings as input and the match ratings as labels. A more sophisticated approach would be to take all the algorithms, combine their results and to also take into account the additional features such as word length or complexity/commonness. Possibly, it could be enough to just use the algorithms that are best for a given word length. However, since the algorithms' results and the additional features (and even the boolean labels as 0 and 1) are numerical values, I could e.g. test the performance of a support vector classifier, or I could try how competitive a neural network approach might be.

Benchmark Model

The performance of my solution could be compared to the results of other plain string distance algorithms (with fixed thresholds), simply by comparing the classification results. In particular, a comparison to similar solutions that are available might be interesting. I have been looking for "fuzzy" solution comparison within different Learning Management Systems (LMSs), but so far I have only found one: Vips for Stud.IP.

Vips offers to use one of two algorithms for identifying close answers to a correct solution. It does not present options for fine-tuning. Both algorithms rely on the implementation found in PHP. Number one is based on the Levenshtein distance that I also implemented for H5P. Number two uses the Soundex algorithm that's based on English phoenetics. The latter seems rather surprising, because Stud.IP is only used in Germany (cmp. MindWires LLC, 2016, p. 7 for an overview of LMS usage in Europe).

Evaluation Metrics

I am facing a classifier, so I can quantify the performance using several metrics. The *accuracy* would tell me the fraction of correct predictions - and it is quite easy to interpret. On the other hand, there's the accuracy paradox which can be misleading, so the *F1 score* would be my metric of choice.

Project Design

The first step of every machine learning project should be to **define the problem**. This proposal reflects the first part of this task, but there may be more to be done. If this proposal is accepted, I am going to delve into some more literature about string distance metrics and their applications in order to find out if there are some more data that might help me on my way.

Step two will be to **collect the data**. This is a project in itself that will not only involve programming a website and think about clever ways to handle the data.

It will also be necessary to promote the website and maybe even manage a little community. However, since such a dataset might be interesting for science, and since the final classifier might be interesting for H5P users and other creators of educational software, I hope there might be some multiplicators to help me out. And who knows, maybe there's even a business application that might profit from my work?

As step three I am going to **analyze and prepare the data**. I may not have to check the data quality on a syntactical level, because sanitization would be done by my own data collection algorithms. Still, there might be need for adjustments. I will have to make sure to remove nonsense data that some jesters could enter for whatever reason. After that, it is crucial to understand the data before experimenting with algorithms. By visualizing the data in different ways, I hope to get some more insight into the problem. This will probably also involve a principal component analysis to filter for features that might not be relevant and thus speed down the algorithms. For example, I could imagine that word length and word complexity might correlate to a large extend.

Step four is to **select algorithms**. I will not only add some more string distance metrics (e.g. Soundex mentioned above), but I will also consider different methods and approaches that might be appropriate for building a classification model. Althought quite a lot algorithms might suit the problem, the right choice will depend on different criteria, such as efficiency and scalability.

Step five will be to **run and evaluate the algorithms** that I have chosen in step four. By comparing each algorithm's performance using e.g. k-fold cross validation, I want to identify the algorithms that are best at "understanding" the data and at attacking the match making.

As step six I am going to **improve and finalize the results** with focused experiments and fine-tuning. Each algorithm has certain parameters that can be tweaked in order to get better results. Approaches such as a grid search can be helpful to support this task systematically. As mentioned before, even ensemble methods could be tried out for improving the results.

I intend to use a Jupyter Notebook to document all these steps including meanders and dead ends.