Scientific article summary

Adam Petříček

With modern websites being extremely complex in terms of providing new possible ways to create features, the code describing these websites is also becoming longer and takes more time to read. While this may not be a problem for humans reading it (it can actually help them to understand the code), it might cause some problems to machines trying to extract some actual piece of information from the code. However, there are some convenient ways to overcome this problem and make text extraction more reliable and less dependent on well-written code. The whole concept is named WFFTE (Webpage multi-feature fusion text extraction).

The first step when extracting text should be getting rid of useless noise with a method called denoising. These are typically scripts, styles, ads, comments and non-relevant information. With this step executed correctly, analyzed code is much smaller and easier to work with. Code is usually filtered using regular expressions to remove designated tags, such as <head>, <script>, <!--> or <style>.

A general way to divide web pages in this context might be to categorize them as single-body and multi-text web pages. Single-body web pages have one container tag with a lot of punctuation marks. On the other hand, multi-text web pages have text distributed across multiple container tags (usually with the purpose of applying different styles to different parts of text). Using this knowledge, we can efficiently construct a DOM tree of the webpage, which is effectively a graph displaying tag hierarchy. Root of the tree is <html> tag, and then each tag is connected to all the tags in contains.

When a DOM tree is successfully constructed, we need to calculate a property called text support for each node in the tree. Text support is calculated with the knowledge of distance from the title, descriptive language for the title and relation between body punctuation, hyperlink text and normal text. Using the calculated result for each container, we can estimate which container is most likely to be the body, because its tag is usually closer to the title than others. It's also useful for the container to include website title text. Body typically has more punctuation in it, this knowledge can also help the estimation.

The whole formula is SD = DSD * (TSD + PSD), where SD = text support, TSD = title support, PSD = general support. Individual parts of the formula are calculated first.

DSD = 1 / (rdi * qi), where rdi = serial number on the label path and q is an integer greater than 0.

TSD = λ * FW + β * SW, where FW = first word and SW = second word. These are the two most common occurring words in container tags. Alpha and beta are constant coefficients helping to balance the calculated text support. Alpha should be lower than beta to avoid unwanted results.

PSD = FP * (NC | HC), where FP = punctuation support, NC = number of words of unlinked text, HC = number of words of the linked test.

Important part of the algorithm is also merging tags with tags containing the same styles or properties located next to each other, to even reduce the code length further.

Having all these values calculated, we can now traverse the elements in the dataset and extract desired contents with higher efficiency. Provided researches are showing this method having averagely 95% accuracy of text extraction, which is few percent higher than other available methods. However, the percent results are not the best possible ones, as they vary depending on the number of input samples. The method is based on large-scale training. Extracting text from multi-text web pages is slightly worse due to the fact of having to deal with more noise.