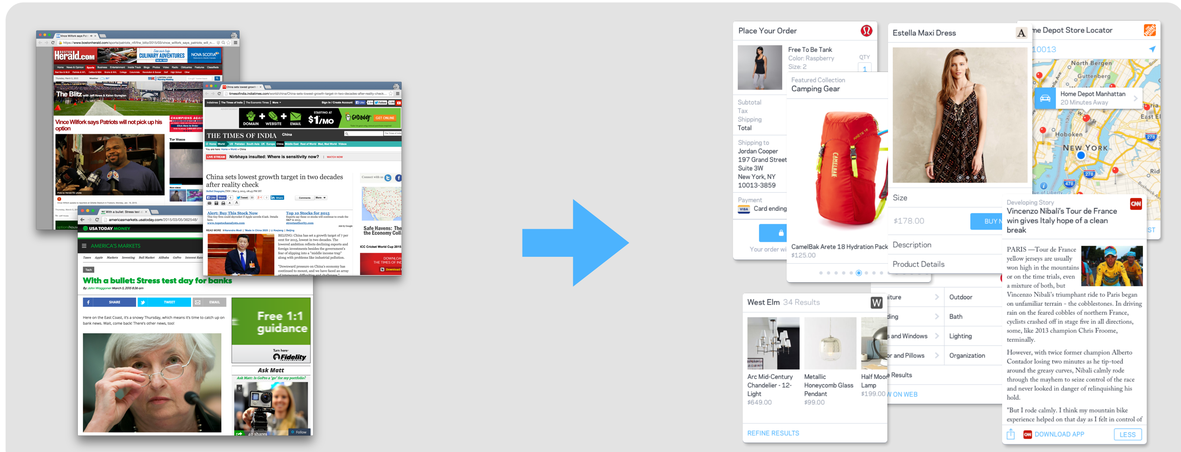
ML Architecture for Content Extraction at Wildcard and an Experiment with a Deep Model

Sven Kreiss

Problem

At Wildcard we think about technologies for a future native mobile web experience through cards. Cards are a new UI paradigm for content on mobile for which we schematize unstructured web content. Part of the challenge is to develop an understanding of online content through machine learning algorithms. The extracted information is used to create cards that are surfaced in the Wildcard iOS app and in other card ecosystems.



Formulation

We are looking for a vector of labels (*title*, *body*, *author*, etc.) for elements on the page, $\vec{l}$, which depend on the document $\mathcal{D}$:

$$\vec{l} = \vec{l}(\mathcal{D})$$

The document can be divided into *s* page segments:

$$\mathcal{D} = (d_1, \ldots, d_s)$$

A particular classification output for page segment *n* can be written as

$$l_n = l_n(l_1, \ldots, l_{n-1}, l_{n+1}, \ldots, l_s, d_1, \ldots, d_s)$$

which makes the dependence on other classification outputs and the entire document explicit. A possible approximation is:

$$l_n = l_n(l_{n-1}, l_{n+1}, d_n, \vec{\theta})$$

which introduces the latent variables $\vec{\theta}$ which absorb the other dependencies.

Features

The features that are directly extracted from the document can be grouped into these four categories:

t**ext**: The most important words for each category from the visible text and the fraction of capital letters, numbers and special characters.

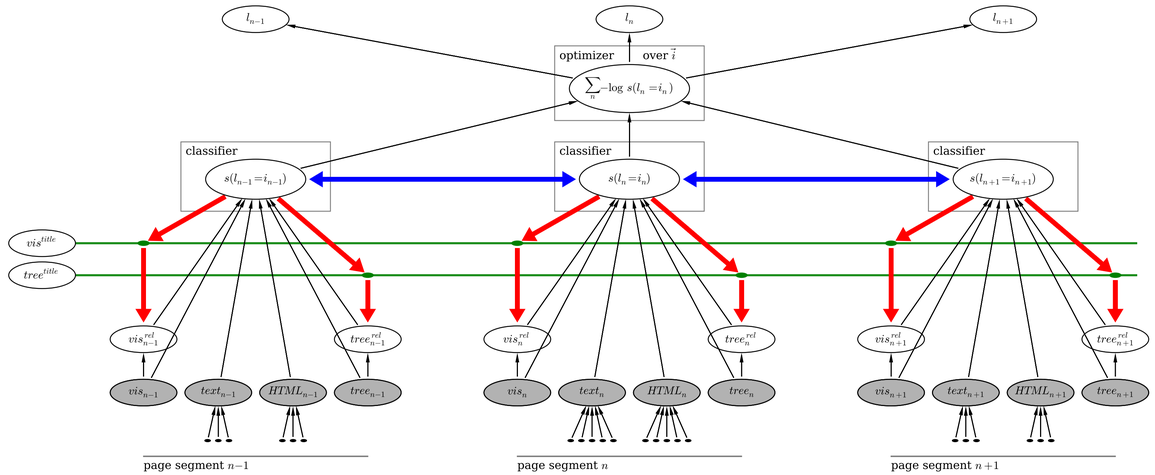
HTML: The top words of the tokenized versions of the HTML tags and selected attributes.

tree: Based on a “Content Tree” (reduced tree structure compared to HTML DOM). Features include tree distance triplets (#up, #sideways, #down) to root and title nodes.

vis (optional): slow to extract visual features using browser emulation that include CSS attributes and the visibility, position and size. Once the *title* is identified, relative information is included (e.g. how much smaller is the font size).

Other inputs to the classifier come from latent variables and neighboring classifications.

Solution: SIC



Unrolled block diagram for sheet 1: classifies *title*, *body*, *author*, *date published*, etc. Shaded nodes are directly extracted from the document and white nodes are inferred.

The input is a set of features for each page segment *n* (a segment is a `<div>`, `<p>`, etc). The output is a label *l* for each segment tagging it as *title*, *body*, *author*, etc.
The inputs are processed by the classifier which calculates a score *s(l)* for each label *l* for each *n*.

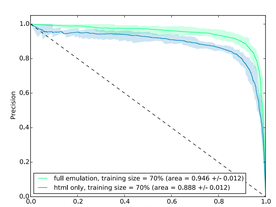
red path: The labels feed back to the feature level and determine the set of features for the best title, vis^{title} and $tree^{title}$. Those are used to calculate relative features vis^{rel} and $tree^{rel}$ (e.g. how much smaller is the font size).

blue path: the classification outputs of neighboring segments are used as input features.

Optimizer: minimizes the total score by scanning over classification outputs i_n. In practice, the classifier produces a first guess without the red and blue paths in a feed-forward manner and then fully reevaluates all classifications at each subsequent optimization step.

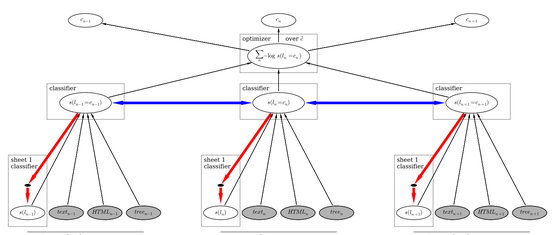
Results and Plans for a Deep Model

Adding the red and blue paths increased the precision, for example, for the *body* label from **82% to 91%**.



Precision-Recall curves for body label (Preliminary): Comparing with and w/o using expensive visual features from browser emulation.

Segments that are classified in sheet 1 as *title*, *body*, *author*, etc. need to be grouped into higher level classes *c* like *article* or *product description* in sheet 2. Sheet 2 also arbitrates which *title* is the relevant one for each segment. The “tools” in the form of blue and red paths, classifiers and optimizers are the same as in sheet 1. Sheet 2 is in development.



Block diagram for sheet 2: classifies *articles*, *products*, etc.

Contact: @svenkreiss sven@trywildcard.com