Tag Prediction in the TiddlyWiki

Joe Armstrong

October 18, 2018

Contents

1 Introduction

This note describes two methods for predicting tags from the content of a tiddler:

Baysian Inference The first method uses a two pass algorithm to predict the tags. In the first pass we record the conditional probabilities for the words in a tiddler given a specific manually assigned tag.

In the second pass, we throw away the tags and predict the tags based on the contents of the tiddler

From the content alone tags are predicted with a roughly 80% accuracy. By this I means that if a tag is predicted there is an 80% chance that is the same as one of the manually assigned tags.

TF*IDF The second method totally ignore the manually assigned tags but just performs a statistical analysis of the content of the tiddlers. Keywords are words in the content that have a high probability of occurrence in the tiddler content and a low overall probability.

2 Step 1 - obtaining the training data

I used the data from `https://tiddlywiki.com/`

I'll assume you have fetched the tiddlywiki and saved this in a file called `tiddlyWiki.html`.

To extract the individual tiddlers I used the node js implementation of the tiddly wiki (available from `http://github/`.

The following script unpacks the individual tiddlers into a sub-directory called `tiddlers`. This gives us a set of 994 tiddlers to work with.

```
#!/bin/sh

outdir=tiddlers
mkdir -p $outdir
in=tiddlywiki.html
tiddlywiki --load $in --output $outdir \
           --render "[!is[system]]" \
           "[encodeuricomponent[]addsuffix[.tid]]" \
           "text/plain" "$:/core/templates/tid-tiddler"
```

The format of the tiddlers is not quite appropriate for this experiment, so the first step is to parse all the tiddlers, extract the content, titles and tags and dump this to a file for future analysis.

3 Bayesian Inference

Bayes theorem is usually written as:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

The notation $P(A \mid B)$ means the probability that `B` has occurred given that `A` has happened. $P(X)$ is the probability that X has happened.

Bayes theorem gives us a simple way to predict which tags a untagged tiddler should have. Given a training pass we can compute $P(word|tag)$ which is probability that a particular word with a given tag exists.

To predict which tags an untagged tiddler should have we compute $P(tag|word)$ for all known tags and choose the value which maximizes this expression summed over all words in the tiddler.

This might sound rather abstract, so I'll start with a simple example.

3.1 Training the classifier

Assume we have 5 training sentences - we call the classifier five times as follows:

```
teach(<<"cars">>,<<"volvo makes many cats cars">>),
teach(<<"cars">>,<<"fantastic miles per gallon">>),

teach(<<"pets">>,<<"black cat called zorro">>),
teach(<<"pets">>,<<"another daft cat called daisy">>),
teach(<<"pets">>, <<"funny dog called sam">>),
```

The first argument to `teach` is a tag, the second is a nonsense text.

Now we can calculate some probabilities.

First we'll calculate $P(cat|pets)$ this is the probability that the word *cat* occurs in a sentence with tag *pets*.

The word *cat* occurs 2 times in sentences with the tag *pets* and there are a total of 13 words in sentences with the tag *pets* so:

$$P(cat|pets) = 2/13$$

What about $P(cat)$? the word *cat* occurs a total of 3 times and there is a total of 22 words so,

$$P(cat) = 3/22$$

Finally $P(pets)$ the tag *pets* occurs 3 times and there are 5 tags so

$$P(pets) = 3/5$$

Now we can evolve Bayes theorem namely

```
P(pets|cats) = P(cat|pets) * P(pets) / P(cat)
             = (2/13)      * (3/5)   / (3/22)
```

3.2 Making a Prediction

To predict which tags a tiddler should have, we iterate over all tags and all words in the tiddler of interest computing the probability that the tag should occur in the tiddler. In pseudo code:

```
for tag in allTags
   P[tag] = 0
   for word in thisTiddler.Content
      p[tag] += P(word|tag)
   end
end
sort the P array and choose the
first K elements
```

For each tiddler I have set $K = 3$ to choose the 3 most likely tags.

4 TF*IDF

`TF` stands for *Term Frequency* and `IDF` for inverse document frequncy.

The Inverse document frequency (IDF) of a word `W` is defined as `log(D/N+1)` where `D` is the total number of tiddlers in the tiddlyWiki and `N` is the number of tiddlers that contain the word `W`.

Example: Suppose we have a tiddlyWiki of 1000 with tiddlers. Assume the word `machine` occurs in 50 tiddlers. The IDF of the word `machine` is thus `log(1000/51)` (= 1.29). If the word `machine` occurs 10 times in a tiddler of 100 words then the TF of `machine` is 10/100 = (0.1) and the TF*IDF weight of the word is 0.129.

The words in a tiddler with the highest TF*IDF scores are chosen as keywords to represent the content.

As in the previous example we compute the 3 most likely tags for each tiddler.

5 Results

The analysis program produces a listing file. Here are a few typical lines from this file:

```
Title: ClearPasswordCommand
    Manual Tags: [<<"Commands">>]
    Predicted Tags: [<<"Commands">>,<<"TableOfContents">>,<<"Editions">>]
    TF*IDF Keywords: [<<"clearpassword">>,<<"language">>,<<"help">>]
Title: Code Blocks in WikiText
```

```
    Manual Tags: [<<"WikiText">>]
    Predicted Tags: [<<"WikiText">>,<<"Operator Examples">>,
                     <<"Filter Operators">>]
    TF*IDF Keywords: [<<"pre">>,<<"monospaced">>,<<"backticks">>]
Title: CodeBlockWidget
    Manual Tags: [<<"Widgets">>]
    Predicted Tags: [<<"Commands">>,<<"Widgets">>,<<"Filter Operators">>]
    TF*IDF Keywords: [<<"language">>,<<"codeblock">>,<<"code">>]
Title: CodeMirror Plugin
    Manual Tags: [<<"OfficialPlugins">>,<<"Plugin Editions">>]
    Predicted Tags: [<<"Plugin Editions">>,<<"OfficialPlugins">>,
                     <<"Resources">>]
    TF*IDF Keywords: [<<"codemirror">>,<<"prerelease">>,<<"latest">>]
```

`Title:` is the the tiddler title. `Manual Tags:` are the manuualy assigned tags. `Predicted Tgaas:` are the tagsprediuced by Bytean infernce. `TF*IDF Keywords:` are the TF*IDF prefdicted keywords.

Iterestingly there are several tiddlers with no predicted tags.

In our data set there were 994 tiddlers and 1437 tag assignments. The total number of different tags was 233.

83% of all tiddlers were correctly tagged by the Bayesian tagger. By this I mean that the tags in the tiddler were found in one of the three most likely predicted tags.

The TF*IDF tags correspond to the manually assigned tags in about 20% of the tiddlers. But this does not mean that the tags are useless.

The problem with the manual tags and the corresponding Bayesian inference has to do with the quality of the tags. Often the tags are assigned for internal purposes, for example to enable high-level transclusion operations. Large numbers of tiddlers (in this sample) have the same tag (for example, `Resources`) this might be a good tag for book-keeping purposes, but is not a good tag for describing the content of the tiddler in a meaningful way.

Often the TF*IDF tags look to me to be very well chosen - I imagine the best approach would be interactive. The user could request a set of predicted tags for a tiddler and then chose to accept or reject the suggestions in an interactive manner.