Hashing Trick Notes

Thomas W. Jones

6/10/2018

Sources for these notes:

https://medium.com/value-stream-design/introducing-one-of-the-best-hacks-in-machine-learning-the-hashing-trick-bf6a9c8af1
https://blog.someben.com/2013/01/hashing-lang/

What is the hashing trick for NLP?

Motivation: need a method that...

1. can deal with out-of-vocabulary words,
2. doesn't require model retraining every time new words are added, and
3. is as accurate as possible.

In addition, we don't have infinite RAM to have an infinitely-large vocabulary.

Hash functions map data of arbitrary sizes to data of a fixed size. Typically, hash functions spit out a number, known as a hash.

Hash functions have the following properites:

1. Deterministic - if you feed in the same inputs, you will always get the same outputs.
2. Fixed range - e.g. the range is always between 0 and 1024.
3. One way - if you have a hash, you cannot perform a reverse lookup to get the input.
4. Collision - hash functions may give the same output for different input.

In NLP, hash functions serve to get rid of vocabulary altogether. You can do this, for example, by mapping strings to a binary vector of length 2^{28}. (*How does this happen? I understand conceptually what I would do with such a function. But how would I implement it?*)

A simple, (but bad) hashing function, might map an integer to each letter (range 0 to 25) and 26 to 36 for numbers 0 to 9. Then a string's hash representation (in 1-space) is the sum of the mapped integers. This can get mapped to N-space by setting the j-th entry of an n-dimensional vector to 1. So, if our hash outputs 99, the 99th entry is 1 and all other entries are 0. One could do this word-by-word and then add the vectors up for the whole string.

A good hashing function has a property called *uniformity*. The property of uniformity means that the range of the function should be output roughly uniformly. In other words, if the range of the hash function is 0 to 99, then the applied hash function should output any given value roughly 1% of the time. (*Should it? Wouldn't it make more sense for the range to have the same distribution as the domain? For language, this should preserve Zipf's law. Also, I'm not sure that the "one-way" property of hashing is desireable either...*)

Another desireable property is called *cascading*. This means that a small change in input leads to a big change in output. (*Wait. Why would this be a desireable property at all?*)

If a hashing function has both uniformity and cascading properties, then the chances of collision are independent of how frequently a word is used. (*Again, I am not convinced that this is uniformly good.*)

When using a hash function, every input string is assigned a vector at the time of prediction. Note that new words will still lessen the accuracy of the classifier. But they won't lead to a total failure to predict.

The risk of collision is reduced by chosing your hash function carefully and having a resulting feature vector that is as large as possible. In some cases you may even want collision. For example, you may wish to group

similar words into the same hash. In this case, you may need to bucket these words before hashing, howver. It's unlikely your hashing function will be able to handle this elegantly.

Hashing with Zipf's law allegedly preserves sparsity. (http://arxiv.org/pdf/0902.2206.pdf) However, it's not clear to me that the cascading and uniformity properties will preserve zipf's law.

How might this apply (or not apply) to my project?

I can see how hashing is useful for NLP in general. But I don't see that it is useful for my problem of dealing with different encodings and languages.

I think it would be interesting to use hashing for topic modeling as a means to handle changing vocabulary, as with document classificaiton. Yet, I would want my hashing function to retain most of the properties of natural language to minimize the effect of the hashing function on your result. (i.e. the result should be roughly the same if you did it on the raw words.) It's an interesting research direction, but one that I think is out-of-scope for what I want to do here.