Technology

node.js - because JS is dynamic, has first class functions and closures and node has a huge ecosystem

Interfaces

Storage interface

The purpose of the storage interface is to make storage swappable, therefore it should focus on what the app needs, not on how to fetch it.

General

init

`init(options, function(err))`

Initializes the storage with the specified options.

- options.name - name of the storage
- options.mode - normal or feeding mode, indicates if triplie plans to query the storage or to feed massive amounts of data to it.

Batch operations (batch)

Batch operations module. Queries that request information instead of doing updates or insert must not be run inside a batch.

begin

`begin(function(err))`

Tells the storage that a batch of inserts and updates will follow. This can be safely ignored by the storage layer if its not required.

end

`end(function(err)`

Tell the storage that the batch has ended.

Dictionary functions (dict)

all

```
all(function(err, [words]))
```

Loads the entire dictionary array. Word objects look like this:

```
{
    id: unique
    word: 'word',
    count: 20
}
```

put

```
put(word, function(err))
putMany([words], function(err))
```

Add one or more new words. If they exist only their count is incremented.

get `js get(string): word, get(id): word getMany([strings]): [words] getMany([ids]): [words]` Get one or many words. The operation should be sync which means the data layer should cache all words.

Markov chains model (markov)

next

```
next([ids], function(err, [words]))
```

Given the n-gram [ids], find all the next possible words. The n-gram is guaranteed to have length <= 5

prev

```
prev([words], function(err, [words]))
```

Similar to next() but finds previous words.

put

```
put(ngram[], function(err))
putMany([ngram[], ngram[], ...], function(err))
```

Puts one or more ngrams to the DB. If they do not exist they should be created. If they exists they should be updated with a +1 count.

For put, the n-grams must have exactly 6 words.

Associations model (assoc)

get

```
get(w, function(err, [assocs]))
get([w1, w2], function(err, assoc))
getMany([w, w, w, ...], function(err, [assocs]))
getMany([[w1, w2], [w1, w2], ...], function(err, [assocs])
```

Gets all associations in which the specified words (or pairs)

An association result is an object:

```
{id1, id2, count}
```

where id1 < id2

put

```
put([w1, w2], function(err))
putMany([[w1, w2], [w1, w2], ...], function(err))
```

Puts one or more associations into the DB. If they do not exist they should be created. If they do, they should be updated with a +1 count.