1) Summary of approach
Our approach to solve this HW3 was a holistic end to end view to first understand
the hypothesis and task at hand, rather than follow the instructions in a sequential
fashion. For this reason we investigated what was expected from the homework and
our interpretation was the following:

Main goal: Discover and label the underlying taxonomy of the data provided
(Wikipedia)

Specific goals:
 a) We needed to create data to be processed from a large file
 b) We needed to discover clusters associated with this data
 c) We needed to label (identify) such clusters
 d) We needed to measure how good these clusters represent the actual taxonomy of
the underlying data.

Additionally, the pedagogical goals were:
 i) Learn how to preprocess large amounts of data on a distributed computing
 platform, i.e. map-reduce (hadoop)
 ii) Learn how to interpret step by step the progress in processing the data
 iii) Learn to evaluate clustering data

Given the task at hand we started by understanding what type of data was expected
at each iteration:

1) Preprocessing → raw Wikipedia dump
2) Dictionary → Cleaned dump word count
3) Extract categories → Category label extraction
4) Clustering → k-Means on Dictionary
5) Evaluation → Weighted Max Matching between labels and discovered clusters

Our end-to-end observation of the challenge returned two initial observations that
impeded to accomplish the previously described flow:

a) There were no aggregated category information available in the Wikipedia dump,
only very fine grained category, mostly at the taxonomy tree leaf level. And there
was no [parent,child] tuples available to reconstruct the taxonomy to higher
levels. Therefore, we needed to find a concrete solution to this problem.
b) Given the previous point, executing Maximum Weight Matching at the leaf level
would have implied to work with approximately 600K categories, therefore making
the algorithm O(N^3) impossible to execute in any considerable amount of time.

From a pedagogical perspective, we also discovered 2 facts:
1) Debugging at hadoop was highly complicated, as some users were continuously using jobs with the large Wikipedia (37GB) file, which took almost the complete capacity of the cluster. Therefore, any small debugging job performed on toy files was left usually waiting in a large queue. We assume most of the users who repeatedly were using the cluster were also debugging, but they were doing so in the main document, which was completely unnecessary.

2) If reconstructing the categories of the Wikipedia dump was considered part of the data pre-processing job, this task well underestimated in the time allocated for the project. After a detailed investigation we discovered that the Wikipedia taxonomy is ill-defined in several ways:
   a) It is a forest of trees based on different categories
   b) It is not a DAG
   c) There are multiple formats defining [parent,child] tuples

Given this complexity, reconstructing the main categories in one of the forest graphs took considerable amount of work using rather complicated tree compression algorithms and making many assumptions such as:
   - Choosing arbitrarily the way parents are selected
   - Choosing arbitrarily the [parent,child] tuples

In summary, we believe that the execution of this homework could have benefited from the following supplementary tasks:

1) Provide some way to support debugging at the cluster level (including end-user training on how to create a single node cluster, and having specially dedicated machines for debugging tasks).
2) Pre-extraction of the categories for the Wikipedia dump file. NOTE: Our team provided a sub-optimal solution to the class, which could be used in the next HW iteration if needed.

In the next section we describe more in detail the tasks we performed.

2) Detailed Overall Tasks

Generate document matrix from the raw Wikipedia file
* Tokenize documents and generate a bag of words representation for each document
* Do word counts over all words in the dataset
* Filter high and low frequency words
* Index the word list, and document list to create a term-document matrix
* Create a list of leaf categories assigned to each document (Or use the resource from dbpedia)
* Map the Category graph using data pulled from the Category Wikipedia entries. (Or use the resource from dbpedia.)
Analysis
At this point the data set is now small enough to analyze on a relatively powerful single computer.

Clustering:
* Aggregate the categories from the category graph and use this to create a mapping from document categories to a small number of high level categories.
* Cluster the documents with the same number of clusters as these high level categories

Evaluation:
* Compute $|d \in D$ documents with labels $W_i$ and $C_j|$ where $W$ is the set of high level Wikipedia categories and $C$ is our cluster assignment. This gives us a measure of the correctness of our cluster assignments compared to the clusters they are members of in Wikipedia.
* Compute matching of Wikipedia clusters to our clusters in matlab using the Hungarian algorithm using the above as edge weights.
* Compute a performance score by computing the percentage of correctly clustered documents.

3) Detailed Suggestions for Students in the Future

Be able to run hadoop on your own “cluster”:
* Even if we have access to a functioning cluster the time it takes to transmit binaries and run in a remote environment inhibits productivity.
  * [http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/](http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/) has a nice tutorial that we were successful in implementing on one machine. Oddly enough, the installation didn’t function on another Ubuntu machine. I think a VM image with a version of the hadoop install similar to the cluster would help tremendously with load and debugging issues.

There is already a mapping from articles to categories available at dbpedia
* This would have saved us one complete hadoop job getting articles from categories: [http://downloads.dbpedia.org/3.7/en/article_categories_en.nt.bz2](http://downloads.dbpedia.org/3.7/en/article_categories_en.nt.bz2)
  * The same site has a copy of the graph hierarchy called the SKOS set: [http://downloads.dbpedia.org/3.7/en/skos_categories_en.nt.bz2](http://downloads.dbpedia.org/3.7/en/skos_categories_en.nt.bz2)
  * The above must be used to generate Wikipedia category assignments as the given article category assignments must be aggregated unless an incredible large number of sparsely populated clusters > 600,000 are going to be created using k-means clustering.

Some other programming resources were of use
Processing the Raw Wikipedia Data:
* An XML parser: [http://xmlandhadoop.blogspot.com/](http://xmlandhadoop.blogspot.com/) should be used to properly partition inputs to each mapper.
* The dataset contains a number of articles that should be ignored. Notably, that redirects are included as well as descriptions for Wikipedia categories.
* It was suggested that a Wikipedia tokenizer be used, which is the one produced by the Lucene project: http://lucene.apache.org/core/. However, there are some other tools that are useful from the same organization such as the stemming and filtering classes.
* The above resource makes the task of tokenizing the documents and converting them into documentName,[(word,wordCount)] tuples much easier.
* The above transformation cuts down the dataset size dramatically making the remaining tasks feasible on a single machine.

**Resources for the final evaluation**

There are two steps in the evaluation process.

First, optimistically label the clusters seen in your dataset with your chosen target Wikipedia cluster, this is done by performing a bipartite matching on a graph where each node represents a cluster. The first set is determined by aggregating the category labels given to each article and your algorithm determines the second set.

* We can then run the Hungarian algorithm on the graph. Given that it is an O(N^3) algorithm, where N is determined by our cluster size, we must keep the number of clusters low.
* Several implementations are available in matlab:
  http://www.mathworks.com/matlabcentral/fileexchange/6543 was the one we planned on using.

Secondly we must compute a statistic measuring the number of document pairs in the same Wikipedia category as in the clusters that we’ve created.

* Given the above scheme there are two obvious statistics used to evaluate the clustering
  * Consider the number of correctly classified articles in aggregate in relation to the total number of articles.
  * On a per cluster basis, use a F1 measure.

**Conclusion**

In conclusion, our holistic end-to-end approach was both a blessing and a curse. By understanding correctly the complete task at hand, we were able to detect important limitations in the type of data available, and the resources needed to compensate for these issues. On the other hand, the time spent on laying down the pieces necessary to solve the complete puzzle limited severely the amount of time left to process data. The latter problem was furthermore exacerbated by the fact that the cluster did not have the best setting to support heavy usage and which did not have a correct policy for debugging.

Overall, we believe the doing data mining in general is very much as much theory as art. It demands not only a great deal of knowledge of different elements of computing, from complexity theory and parallel computing, to notions of machine learning, but also a very good deal of practice, in order to identify the right amounts of resources needed, the types and sizes of data elements that should be processed
at different stages and the overall assessment of an end-to-end challenge in terms of time and resources. Additionally, the right combination of the many degrees of freedom available to test the best algorithm configurations demands not only some expertise, but enough time to generate the many computations needed.

We believe therefore that it is important to provide future students with two key elements of knowledge: 1) an intuition and clear understanding of the type of hypothesis to be tested, developed before and during the mining process, in order to avoid a “brut force” approach, based on trying to generate data without a clear goal in mind (i.e. avoid the “shit in, shit out” problem); 2) a good notion of the overall project complexity by adding some data pre/post-processing and project management heuristics that could be used to improve the potential outcome of a complete “big” data project.