Machine Learning Coding Challenge

**Problem Definition**

As the label provided in both training and testing set are not reliable, so I treat this problem as an unsupervised ML problem, thus in the following sections, the label will not be used.

**Challenge**

As I don’t know how many topics this corpus could have, so most important thing is to determine number of topics first. In the following section, I applied two methods to bring some insights.

**Method**

1. Data preprocessing:
   1. Random sample 10% from both training and testing set
   2. From sample data in above, applied text cleansing including, tokenize, stop words removing, lemmatization
   3. Output two files with raw data, raw data label and the cleaned content
2. LDA model: In natural language processing, the latent Dirichlet allocation (LDA) is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar.
   1. Determine topic number: as the sampled set still cost time to run, so from this sample, I randomly choose 500 rows from training set and applied a grid search under different number of topics for the purpose of comparison two metrics, which are coherence and perplexity respectively. From the perplexity (the smaller, the better) plot on the right, there is no obvious elbow point for us to choose. Also, when I check the coherence (the larger, the better), it still does not provide much clue for determining the topic number, so I chose 15 as my best guess of topic number for further analysis.

A close up of a logo

Description automatically generatedA screenshot of a cell phone

Description automatically generated

* 1. From above model, I can interpret several topics. For example, as can be seen in the image below, the first topic seems very likely to be health insurance as ‘cancer’, ‘healthy’ and ‘insurance’; While

A close up of a newspaper

Description automatically generated

1. Word2Vec + Kmeans
   1. By training a wrod2vec model based on all training data, I used the average value of each word vectors for represent that given content, as document representation, and then applied K-means on document vectors
   2. Again, there is no obvious clue for me to choosing number of clusters, I compare inertia (the smaller, the better) for each cluster which shown as below,

A screenshot of a cell phone

Description automatically generated

* 1. Thus, I choose 15 as number of clusters just like the decision made on LDA model.
  2. However, it seems that keywords for each cluster centres have not provided very obvious clue to draw any topic, which can be seen as below,

A close up of a newspaper

Description automatically generated

**Conclusion**

Above sections explained how I think of this problem and what method I can try for solving it, in this section, I will make conclusion along with all the answers to extra questions in requirement.

* How do you evaluate the accuracy and correctness of your model(s)?

Answer: There are two part for evaluate correctness of model,

* + Model metric, i.e., perplexity / coherence in LDA model, and inertia to evaluate K-means model, however, as there is no baseline, so this result can be just used as baseline.
  + Intuition of the topic clustered by model, i.e., from the most weighted keywords in each topic/cluster, how easily human being can determine what topic it is.
* What could you do to improve the accuracy of your model?

Answer: There are some issues within the current solution

* + Data clean issue: from what I observed in the top 15 topic model keyword and top 15 k-means centres’ keywords, there are many keywords, such as ‘australia’, ‘sydney’, ‘law’, provide very less information but with relatively higher frequency to be picked up across many topics/clusters, thus, solution to this are
    - Gradually increasing stop words list, e.g., ‘sydney’, ‘melbourne’, or some verb/adj/adv without any mean, such as ‘get’
    - Add bi-gram or even tri-gram as a whole word for being used as keywords feature as well
  + Try to transfer it to a supervised learning problem. Specifically, maybe I can label samples with reasonable size by myself or crowed-source company, then I can transform this problem as like text classification problem or ranking problem (e.g., using Siamese Network)
* How would you serve your model(s) in production?

Answer: There are few steps

* + Save the model as pickle or some other types of format
  + Write up requirement including details in each step, including data pre-processing, modelling calling function, metric monitoring and so on
  + Work with engineering team to implement it (this is normally what happen in my position), but I think I can also wrap it as micro-service by using flask (this is what I have tried so far)
* Imagine that there are future requirements that ask you to add a new class/topic/pattern? What would re-training look like? Are there any concerns? How would you avoid these concerns?

Answer: I think it will very depend on specific situation. In terms of the re-training process, it will be looks like I need to analysis what the problem actually is, for example, does this new class/topic really a new class/topic, or just a subtopic belongs to its father topic. As if so, I may do not need to retrain the whole model, but just add another sub-model make classification within a given group. However, if I do need to retrain a model, then I need to well define the problem and model it directly, several concerns I can think of list as below,

* + New class’s sample is not enough (size might very small) which cause imbalance class issue: I might need to use reasonable sampling method (up-sampling, down-sampling or data argument methods)
  + New patterns might evolve over time frequently which will very cost to retrain the whole model, what I can think of to avoid is that we validate how large improvement of this pattern could add the existing model by retrain on different samples with reasonable size over past time, if it very significant, then we continue to investigate more.