# Note On Developing

**Objects**: Analysis (IDF, ModelFunctions), Classify, Token(Stopwords), DocGathering

**Help libraries custom:**

**Classes:** IDFValue, Dictionary, DocumentVector, NormalisedVector, TokenizedText <= Tokenizer, DocumentFinder

**Workflow:** One or more directories can be entered, and they can be explored *recursively* to gather every document which has mime type of the like *docx, doc, txt, pdf (work in progress), md (work in progress (my need a library and to exclude hash symbol).* Sub-directories set to be excluded are excluded in the process (helper method *recursiveTreeDocsSearch* to gather all docs in a folder).

**Attempt OOP**

Creation of classes representing vectors, depending on the model wanted they would implement different traits, visibly divided such tf, idf or bag of words.

They contain functions that model the vector upon certain criteria. They always ultimately return (1st iteration of code) a map value Double assign to an index String(Token), which is respectively the weight of the term (tf: importance based on frequency, bag-of-words: frequency) and the term itself. Idf are values brought from outside (they are calculated separately using all documents being analysed as parameters for construction) although they have the same type. (2nd iteration of the code) a tuple consisting of the term and the weight. They will be converted to a map on a second moment.

In the second attempt there would be only one vector class which would be fed into a NormalisedVector class to normalised its length in relation to all other documents, letting normalisation easing up the process of comparison. This normalised vector would be then specialised over specific type depending on the model sought, which would be achieved with subtypes implementing different functionalities (different traits).

**Attempt Functional**

As second OOP attempt, removing any subtype and let the modelling being fed into the NormalisedVector as a function. This could be any process as long as from a tokenised document and a term would produce a tuple of a term and a decimal value.

This could give more flexibility over the model that can be achieved, and the user/future developer could feed in any function with those criteria to achieve the wanted modelled vector.

The vector would still be an entity per-se to group within it more information such the size of it, and the path to the original document.