**Extractive Summarization:**Extractive methods attempt to summarize articles by selecting a subset of words that retain the most important points.

This approach weights the important part of sentences and uses the same to form the summary. Different algorithm and techniques are used to define weights for the sentences and further rank them based on importance and similarity among each other.

**Input document → sentences similarity → weight sentences → select sentences with higher rank.**

I will be using an [**unsupervised learning**](https://en.wikipedia.org/wiki/Unsupervised_learning)approach to find the sentences similarity and rank them. One benefit of this will be, you don’t need to train and build a model prior start using it for your project.

**Cosine similarity** is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. Since we will be representing our sentences as the bunch of vectors, we can use it to find the similarity among sentences. Its measures cosine of the angle between vectors. Angle will be **0**if sentences are similar.

Input article → split into sentences → remove stop words → build a similarity matrix → generate rank based on matrix → pick top N sentences for summary.

We have used **Textrank**as an approach to rank the sentences. Text Rank does not rely on any previous training data and can work with any arbitrary piece of text. Text Rank is a general-purpose graph-based ranking algorithm for NLP.

The **TextRank** Model Graph-based ranking algorithms are essentially a way of deciding the importance of a vertex within a graph, based on global information recursively drawn from the entire graph.

The basic idea implemented by a graph-based ranking model is that of “voting” or “recommendation”. When one vertex links to another one, it is basically casting a vote for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex.

Moreover, the importance of the vertex casting the vote determines how important the vote itself is, and this information is also taken into account by the ranking model. Hence, the score associated with a vertex is determined based on the votes that are cast for it, and the score of the vertices casting these vote

The TextRank keyword extraction algorithm is fully unsupervised and proceeds as follows.

* First, the text is tokenized, and annotated with part of speech tags – a preprocessing step required to enable the application of syntactic filters
* All lexical units that pass the syntactic filter are added to the graph, and an edge is added between those lexical units that co-occur within a window of words
* After the graph is constructed (undirected unweighted graph), the score associated with each vertex is set to an initial value of 1, and the ranking algorithm is run on the graph for several iterations until it converges – usually for 20-30 iterations, at a threshold of 0.0001
* Once a final score is obtained for each vertex in the graph, vertices are sorted in reversed order of their score, and the top vertices in the ranking are retained for post-processing