What are N-Grams?

N-grams of texts are extensively used in text mining and natural language processing tasks. They are basically a set of co-occuring words within a given window and when computing the n-grams you typically move one word forward (although you can move X words forward in more advanced scenarios). For example, for the sentence *"The cow jumps over the moon"*. If N=2 (known as bigrams), then the ngrams would be:

- the cow
- cow jumps
- jumps over
- over the
- the moon

So you have 5 n-grams in this case. Notice that we moved from the->cow to cow->jumps to jumps->over, etc, essentially moving one word forward to generate the next bigram.

If N=3, the n-grams would be:

- the cow jumps
- cow jumps over
- jumps over the
- over the moon

So you have 4 n-grams in this case. When **N=1**, this is referred to as **unigrams** and this is essentially the individual words in a sentence. When **N=2**, this is called **bigrams** and when **N=3** this is called **trigrams**. When N>3 this is usually referred to as four grams or five grams and so on.

How many N-grams in a sentence?

If X=Num of words in a given sentence K, the number of n-grams for sentence K would be:

What are N-grams used for?

N-grams are used for a variety of different task. For example, when developing a language model, n-grams are used to develop not just unigram models but also bigram and trigram models. Google and Microsoft have developed web scale n-gram models

that can be used in a variety of tasks such as spelling correction, word breaking and text summarization. Here is a publicly available web scale n-gram model by Microsoft: **http://research.microsoft.com/en-us/collaboration/focus/cs/web-ngram.aspx**. Here is a paper that uses Web N-gram models for text summarization:***Micropinion Generation: An Unsupervised Approach to Generating Ultra-Concise Summaries of Opinions***

Another use of n-grams is for developing features for supervised Machine Learning models such as SVMs, MaxEnt models, Naive Bayes, etc. The idea is to use tokens such as bigrams in the feature space instead of just unigrams. But please be warned that from my personal experience and various research papers that I have reviewed, the use of bigrams and trigrams in your feature space may not necessarily yield any significant improvement. The only way to know this is to try it!

Java Code Block for N-gram Generation

This code block generates n-grams at a sentence level. The input consists of **N** (the size of n-gram), **sent** the sentence and **ngramList** a place to store the n-grams generated.

```
private static void generateNgrams(int N, String sent, List ngramList) {
  String[] tokens = sent.split("\\s+"); //split sentence into tokens

  //GENERATE THE N-GRAMS
  for(int k=0; k<(tokens.length-N+1); k++){
    String s="";
    int start=k;
    int end=k+N;
    for(int j=start; j<end; j++){
     s=s+""+tokens[j];
    }
    //Add n-gram to a list
    ngramList.add(s);
  }
}//End of method
```

Online API for N-gram Generation

Here is a **Web API** for on demand word count and N-Gram Generation