1) Difference between Lovins stemmer and Porter stemmer are:-

Porter stemmer -

- It is based on the idea that the suffixes in the English language (approximately 1200) are mostly made up of a combination of smaller and simpler suffixes. It has five steps, and within each step, rules are applied until one of them passes the conditions. If a rule is accepted, the suffix is removed accordingly, and the next step is performed. The resultant stem at the end of the fifth step is returned.
- The rule looks like the following:
    - <condition> <suffix> → <new suffix>
- For example, a rule (m>0) EED → EE means “if the word has at least one vowel and consonant plus EED ending, change the ending to EE”. So “agreed” becomes “agree” while “feed” remains unchanged.This algorithm has about 60 rules and is easy to comprehend.
- It produces less Error than the Lovins stemmer.
- It is a lighter stemmer.
- It took 5 steps for stems to be produced

Lovins stemmer -

- It performs a lookup on a table of 294 endings, 29 conditions and 35 transformation rules, which have been arranged on a longest match principle .
- It removes the longest suffix from the word.
- Once the ending is removed, the word is recoded using a different table that makes various adjustments to convert these stems into valid words.
- It always removes a maximum of one suffix , due to nature of its as single pass algorithm.
- It is very fast as compared to porter stemmer and can handle removal of double letters in words like ‘getting’ being transformed to ‘get’ and also handles many irregular plurals like - mouse and mice, index and indices etc.
- It is more space consuming than porter stemmer.
- Many suffixes are not available in the table of endings .
- It is sometimes highly unreliable and frequently fails to form words from the stems or to match the stems of like-meaning words.
- It is a heavier stemmer that produces a better data reduction.

3) Difference of stemming and lemmatization are :-

The basic function of both the methods − stemming and lemmatizing is similar. Both of them reduce a word variant to its ‘stem’ in stemming and ‘lemma’ in lemmatizing. There is a very subtle difference between both the concepts. In stemming the ‘stem’ is obtaining after applying a set of rules but without bothering about the part of speech (POS) or the context of the word occurrence. In contrast, lemmatizing deals with obtaining the ‘lemma’ of a word which

involves reducing the word forms to its root form after understanding the POS and the context of the word in the given sentence.

In stemming, conversion of morphological forms of a word to its stem is done assuming each one is semantically related. The stem need not be an existing word in the dictionary but all its variants should map to this form after the stemming has been completed. There are two points to be considered while using a stemmer:

- Morphological forms of a word are assumed to have the same base meaning and hence should be mapped to the same stem
- Words that do not have the same meaning should be kept separate

These two rules are good enough as long as the resultant stems are useful for our text mining or language processing applications. For languages with relatively simple morphology, the influence of stemming is less than for those with a more complex morphology. Most of the stemming experiments done so far are for English and other west European languages.

Lemmatizing deals with the complex process of first understanding the context, then determining the POS of a word in a sentence and then finally finding the 'lemma'. In fact an algorithm that converts a word to its linguistically correct root is called a lemmatizer. A lemma in morphology is the canonical form of a lexeme. Lexeme, in this context, refers to the set of all the forms that have the same meaning, and lemma refers to the particular form that is chosen by convention to represent the lexeme.

In computational linguistics, a stem is the part of the word that never changes even when morphologically inflected,

while a lemma is the base form of the verb. Stemmers are typically easier to implement and run faster, and the reduced accuracy may not matter for some applications. Lemmatizers are difficult to implement because they are related to the semantics and

the POS of a sentence. Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes. The results are not always morphologically right forms of words. Nevertheless, since document index and queries are stemmed "invisibly" for a user, this peculiarity should not be considered as a flaw, but rather as a feature distinguishing stemming from lemmatization. Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the lemma.

For example, the word inflations like gone, goes, going will map to the stem 'go'. The word 'went' will not map to the same stem. However a lemmatizer will map even the word 'went' to the lemma 'go'.