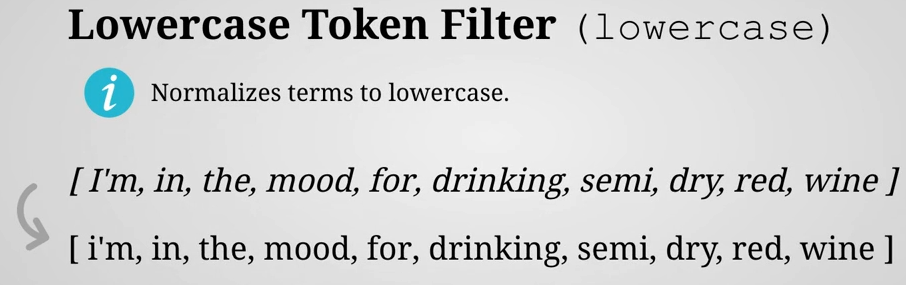
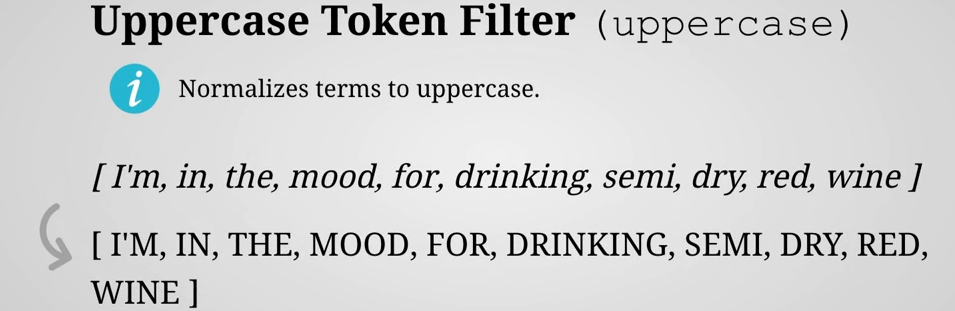
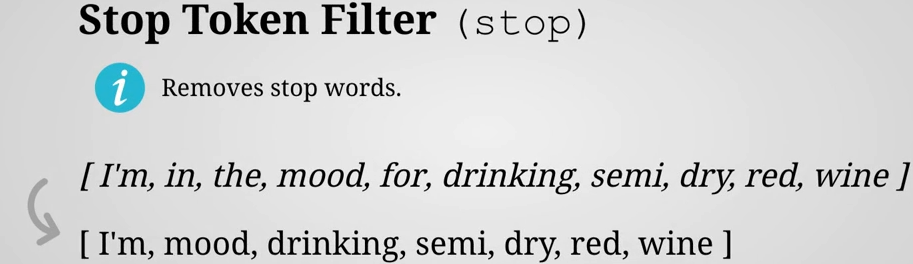
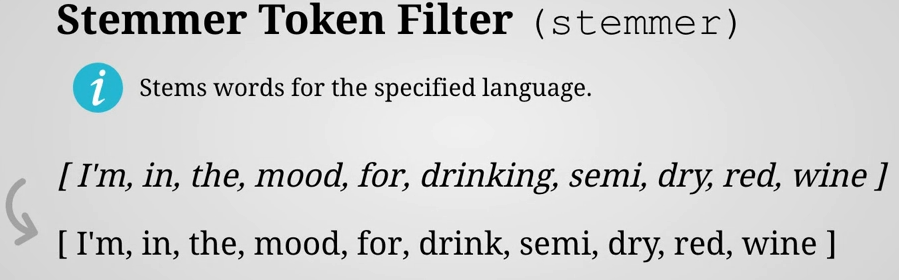
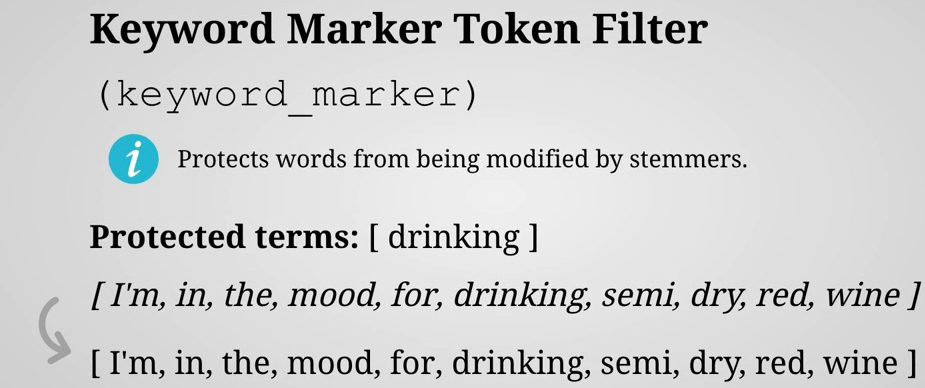
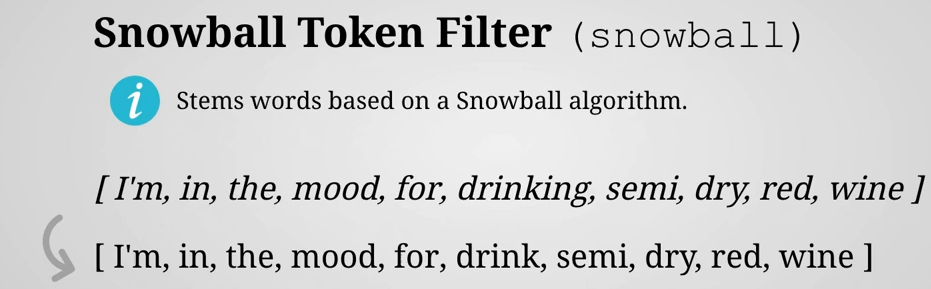
1. ****
2. The next step of an analyzer it’s a token filters.
3. **Agenda**: This lecture is indented to give you a quick overview of the token filters.
4. There are quite a few of them so I will only mention the most important ones and try to be brief just to not take up too much of your time.
5. **Standard Token Filter** (standard)
   1. ****
   2. First up is the standard token filter which doesn’t do anything because it just acts as a placeholder for future versions of Elasticsearch that may need to make use of it.
6. **Lowercase Token Filter** (lowercase)
   1. ****
7. **Uppercase Token Filter** (uppercase)
   1. ****
8. **NGram Token Filter** (nGram)
   1. ****
   2. Actually, we have ngram tokenizer. The behaviour of this ngram token filter is exactly same as that of ngram tokenizer.
   3. **NOTE**: We have the NGram tokenizer too then why this NGram Token Filter?  
      Actually, you can use a different tokenizer instead of using ngram tokenizer then use the ngram token filter.
9. **Edge NGram Token Filter** (edgeNGram)
   1. 
   2. We know that there is Edge NGram tokenizer too. Its behaviour is also same as that of Edge NGram tokenizer.
10. **Stop Token Filter** (stop)
    1. ****
    2. Stop filter removes the stop words (and, or, am etc).
    3. Elasticsearch provides a list of stop words for most languages that can be used.   
       But you have the option of specifying your own configuration if you need to.
11. **Word Delimiter Token Filter** (word\_delimiter)
    1. ****
    2. It splits words into sub-words based on a number of rules which can be enabled or disabled as you please.
    3. It splits into terms at every non-alphanumeric character.   
       It also splits when the case transitions from lowercase to uppercase or vice-versa.  
       It also splits when switching from letters to numbers or the other way around.  
       **NOTE**: The apostrophe followed by s, these two characters are removed.
12. **Stemmer Token Filter** (stemmer)
    1. ****
    2. It stems words based on the specified language defaulting to english.
    3. It reduces words to their base form such as “drinking” to “drink” to make sure that a document is matched regardless of which form a word is in.
13. **Keyword Marker Token Filter** (keyword\_marker)
    1. ****
    2. Protects words from being modified by stemmers. You can do this with the keyword marker filter simply specify a list of terms that should be protected and any stemmer filter will ignore it.
    3. Notice how the word “drinking” has not been stemmed because it was protected from stemmed.
14. **Snowball Token Filter** (snowball)
    1. ****
    2. Another token filter that is used for stemming is “snowball filter”.
    3. It enables you to make use of stemming algorithms implemented in snowball which is a string processing programming language used to implement stammers.
15. **Synonym Token Filter** (synonym)
    1. ****
    2. We want the document to match regardless of which of the words it contains.
    3. That can be accomplished by supplying a configuration file with words that have the same meaning.  
       For instance “happy” and “delighted” when run through a synonym token filter, synonyms will be injected as terms.  
       In the above example, see how “delighted” has been added as a term.  
       The slash is for demo purpose but it is not like that the synonym would be added into the inverted index.  
       The main point is that the synonyms start at the same position as the original term. Why this is important?   
       Because this means that phrase query will still match. That is something that we have not looked at yet but it basically means that you can still write terms that match terms in a specific order and because the order is preserved so that would still work.  
       So the document containing the terms from this example would match both search for “happy” and “delighted”.  
       This helps make all search functionality more intelligent and gives the user the sense that it actually understands the natural language in which the search query is written.
16. These are the tokens filters. Apart from these token filters there are other token filters which are very simple such as
    1. **Trim token filter**: This trims any leading or tailing whitespaces.   
       Although this is typically not necessary as this is handled by most tokenizer.
    2. **Link token filter**: It removes tokens that are either too short or too long based on the minimum and maximum lengths that you specify.
    3. **Truncate Filter**: It is used to truncate tokens into a maximum length that you specify.
17. **NOTE**: For full list of token filters, please refer to the documentation.