Tokenization, Normalization, and Lemmatization

What does it do? Something so simple that it's hard to think of it as a transformation at all: breaks a text into an ordered list of words.

"But isn't a text *already* an ordered list of words?" Unfortunately, not in any sense a computer can understand. You might think that you can break words up at spaces, say; but then you'll end up with words like "`Unfortunately,`" that have a punctuation mark attached.

The big three of pre-processing are:

1. **Tokenization**: splitting words from each other.
2. **Normalization**: Making it so that every word is represented the same. Many choose to **lowercase** every word in a text, for example. If you have archaic spellings or characters, you might want to modernize them. And so forth.
3. **Lemmatization** is a highly optional step that reduces a word to its *stem*. In English, for example, you might reduce "falling" and "fallers" down to the single stem "fall."

What's so great about it?

Not much. It's just a series of hard choices someone will criticize down the road. But it's an absolutely *necessary* set of choices to make other things possible.

The fewer words you have, the more readable many of your models may be. And without tokenization, you can't really do text analysis at all.

What software should I use? In a full pipeline, this is frequently done with libraries for text analysis. For example: Lincoln Mullen has written a tokenization package for R; the `nltk` package for python includes many functions for lemmatizing or tokenizing.

Often, people simply do it themselves using *regular expressions*.

Google around; this is a basic enough set of operations that there are many options.

Are there alternative implementations? Lots. For English, the Penn Treebank tokenization is somewhat standard; that does things like break the word "can't" into the two tokens "ca" and "n't".

The Unicode consortium defines it's own word boundaries. Or you can just use any sequence of characters in a regular expression, which is often the easiest. That regular expression is `[A-Za-z]+` in plain English text, `\p{L}+` in Unicode.