Tutorial: Quickstart

TextBlob aims to provide access to common text-processing operations through a familiar interface. You can treat `TextBlob` objects as if they were Python strings that learned how to do Natural Language Processing.

Create a TextBlob

First, the import.

```
>>> from textblob import TextBlob
```

Let's create our first `TextBlob`.

```
>>> wiki = TextBlob("Python is a high-level, general-purpose programming language.
```

Part-of-speech Tagging

Part-of-speech tags can be accessed through the `tags` property.

```
>>> wiki.tags
[('Python', 'NNP'), ('is', 'VBZ'), ('a', 'DT'), ('high-level', 'JJ'), ('general-pu
```

Noun Phrase Extraction

Similarly, noun phrases are accessed through the `noun_phrases` property.

```
>>> wiki.noun_phrases
WordList(['python'])
```

Sentiment Analysis

The `sentiment` property returns a namedtuple of the form `Sentiment(polarity, subjectivity)`. The polarity score is a float within the range [-1.0, 1.0]. The subjectivity is a float within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective.

```
>>> testimonial = TextBlob("Textblob is amazingly simple to use. What great fun!")
>>> testimonial.sentiment
Sentiment(polarity=0.39166666666666666, subjectivity=0.4357142857142857)
>>> testimonial.sentiment.polarity
0.39166666666666666
```

Tokenization

You can break TextBlobs into words or sentences.

```
>>> zen = TextBlob("Beautiful is better than ugly. "
...                "Explicit is better than implicit. "
...                "Simple is better than complex.")
>>> zen.words
WordList(['Beautiful', 'is', 'better', 'than', 'ugly', 'Explicit', 'is', 'better',
>>> zen.sentences
[Sentence('Beautiful is better than ugly.'), Sentence('Explicit is better than imp
```

`Sentence` objects have the same properties and methods as TextBlobs.

```
>>> for sentence in zen.sentences:
...     print(sentence.sentiment)
```

For more advanced tokenization, see the Advanced Usage guide.

Words Inflection and Lemmatization

Each word in `TextBlob.words` or `Sentence.words` is a `Word` object (a subclass of `unicode`) with useful methods, e.g. for word inflection.

```
>>> sentence = TextBlob('Use 4 spaces per indentation level.')
>>> sentence.words
WordList(['Use', '4', 'spaces', 'per', 'indentation', 'level'])
>>> sentence.words[2].singularize()
'space'
>>> sentence.words[-1].pluralize()
'levels'
```

Words can be lemmatized by calling the `lemmatize` method.

```
>>> from textblob import Word
>>> w = Word("octopi")
>>> w.lemmatize()
'octopus'
>>> w = Word("went")
>>> w.lemmatize("v")  # Pass in WordNet part of speech (verb)
'go'
```

WordNet Integration

You can access the synsets for a `Word` via the `synsets` property or the `get_synsets` method, optionally passing in a part of speech.

```
>>> from textblob import Word
>>> from textblob.wordnet import VERB
>>> word = Word("octopus")
>>> word.synsets
[Synset('octopus.n.01'), Synset('octopus.n.02')]
>>> Word("hack").get_synsets(pos=VERB)
[Synset('chop.v.05'), Synset('hack.v.02'), Synset('hack.v.03'), Synset('hack.v.04'
```

You can access the definitions for each synset via the `definitions` property or the `define()` method, which can also take an optional part-of-speech argument.

```
>>> Word("octopus").definitions
['tentacles of octopus prepared as food', 'bottom-living cephalopod having a soft
```

You can also create synsets directly.

```
>>> from textblob.wordnet import Synset
>>> octopus = Synset('octopus.n.02')
>>> shrimp = Synset('shrimp.n.03')
>>> octopus.path_similarity(shrimp)
0.1111111111111111
```

For more information on the WordNet API, see the NLTK documentation on the WordNet interface.

WordLists

A `WordList` is just a Python list with additional methods.

```
>>> animals = TextBlob("cat dog octopus")
>>> animals.words
WordList(['cat', 'dog', 'octopus'])
>>> animals.words.pluralize()
WordList(['cats', 'dogs', 'octopodes'])
```

Spelling Correction

Use the `correct()` method to attempt spelling correction.

```
>>> b = TextBlob("I havv goood spelingl")
>>> print(b.correct())
I have good spelling
```

`Word` objects have a `spellcheck()` `Word.spellcheck()` method that returns a list of (`word`, `confidence`) tuples with spelling suggestions.

```
>>> from textblob import Word
>>> w = Word('falibility')
>>> w.spellcheck()
[('fallibility', 1.0)]
```

Spelling correction is based on Peter Norvig's "How to Write a Spelling Corrector"[1] as implemented in the pattern library. It is about 70% accurate [2].

Get Word and Noun Phrase Frequencies

There are two ways to get the frequency of a word or noun phrase in a `TextBlob`.

The first is through the `word_counts` dictionary.

```
>>> monty = TextBlob("We are no longer the Knights who say Ni. "
...                  "We are now the Knights who say Ekki ekki ekki PTANG.")
>>> monty.word_counts['ekki']
3
```

If you access the frequencies this way, the search will not be case sensitive, and words that are not found will have a frequency of 0.

The second way is to use the `count()` method.

```
>>> monty.words.count('ekki')
3
```

You can specify whether or not the search should be case-sensitive (default is `False`).

```
>>> monty.words.count('ekki', case_sensitive=True)
2
```

Each of these methods can also be used with noun phrases.

```
>>> wiki.noun_phrases.count('python')
1
```

Translation and Language Detection

New in version `0.5.0`.

TextBlobs can be translated between languages.

```
>>> en_blob = TextBlob(u'Simple is better than complex.')
>>> en_blob.translate(to='es')
TextBlob('Lo simple es mejor que lo complejo.')
```

If no source language is specified, TextBlob will attempt to detect the language. You can specify the source language explicitly, like so. Raises `TranslatorError` if the TextBlob cannot be translated into the requested language or `NotTranslated` if the translated result is the same as the input string.

```
>>> chinese_blob = TextBlob("美丽优于丑陋")
>>> chinese_blob.translate(from_lang="zh-CN", to='en')
TextBlob('Beauty is better than ugly')
```

You can also attempt to detect a TextBlob's language using `TextBlob.detect_language()`.

```
>>> b = TextBlob("بسيط هو أفضل من مجمع")
>>> b.detect_language()
'ar'
```

As a reference, language codes can be found here.

Language translation and detection is powered by the Google Translate API.

Parsing

Use the `parse()` method to parse the text.

```
>>> b = TextBlob("And now for something completely different.")
>>> print(b.parse())
And/CC/O/O now/RB/B-ADVP/O for/IN/B-PP/B-PNP something/NN/B-NP/I-PNP completely/RB
```

By default, TextBlob uses pattern's parser [3].

TextBlobs Are Like Python Strings!

You can use Python's substring syntax.

```
>>> zen[0:19]
TextBlob("Beautiful is better")
```

You can use common string methods.

```
>>> zen.upper()
TextBlob("BEAUTIFUL IS BETTER THAN UGLY. EXPLICIT IS BETTER THAN IMPLICIT. SIMPLE
>>> zen.find("Simple")
65
```

You can make comparisons between TextBlobs and strings.

```
>>> apple_blob = TextBlob('apples')
>>> banana_blob = TextBlob('bananas')
>>> apple_blob < banana_blob
True
>>> apple_blob == 'apples'
True
```

You can concatenate and interpolate TextBlobs and strings.

```
>>> apple_blob + ' and ' + banana_blob
TextBlob("apples and bananas")
>>> "{0} and {1}".format(apple_blob, banana_blob)
'apples and bananas'
```

n-grams

The `TextBlob.ngrams()` method returns a list of tuples of `n` successive words.

```
>>> blob = TextBlob("Now is better than never.")
>>> blob.ngrams(n=3)
[WordList(['Now', 'is', 'better']), WordList(['is', 'better', 'than']), WordList([
```

Get Start and End Indices of Sentences

Use `sentence.start` and `sentence.end` to get the indices where a sentence starts and ends within a `TextBlob`.

```
>>> for s in zen.sentences:
...     print(s)
...     print("---- Starts at index {}, Ends at index {}".format(s.start, s.end))
Beautiful is better than ugly.
---- Starts at index 0, Ends at index 30
Explicit is better than implicit.
---- Starts at index 31, Ends at index 64
Simple is better than complex.
---- Starts at index 65, Ends at index 95
```

Next Steps

Want to build your own text classification system? Check out the Classifiers Tutorial.

Want to use a different POS tagger or noun phrase chunker implementation? Check out the Advanced Usage guide.

[1] http://norvig.com/spell-correct.html
[2] http://www.clips.ua.ac.be/pages/pattern-en#spelling
[3] http://www.clips.ua.ac.be/pages/pattern-en#parser