```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer



nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

```
text = "This is a sample text for NLP processing. It includes tokenization, stemming, and lemmatization."



tokens = word_tokenize(text)
print("Tokens:", tokens)
```

```
Tokens: ['This', 'is', 'a', 'sample', 'text', 'for', 'NLP', 'processing', '.', 'It', 'includes', 'tokenization', ',', 'stemming
```

```
stop_words = set(stopwords.words('english'))
filtered_tokens = [word for word in tokens if word.lower() not in stop_words]
print("Filtered Tokens:", filtered_tokens)
```

```
Filtered Tokens: ['sample', 'text', 'NLP', 'processing', '.', 'includes', 'tokenization', ',', 'stemming', ',', 'lemmatization'
```

```
stemmer = PorterStemmer()
stemmed_tokens = [stemmer.stem(word) for word in filtered_tokens]
print("Stemmed Tokens:", stemmed_tokens)
```

```
Stemmed Tokens: ['sampl', 'text', 'nlp', 'process', '.', 'includ', 'token', ',', 'stem', ',', 'lemmat', '.']
```

```
lemmatizer = WordNetLemmatizer()
lemmatized_tokens = [lemmatizer.lemmatize(word) for word in filtered_tokens]
print("Lemmatized Tokens:", lemmatized_tokens)
```

```
Lemmatized Tokens: ['sample', 'text', 'NLP', 'processing', '.', 'includes', 'tokenization', ',', 'stemming', ',', 'lemmatizatio
```

```
import gensim.downloader as api



glove_model = api.load("glove-wiki-gigaword-100")
```

```
[================================================] 100.0% 128.1/128.1MB downloaded
```

```
glove_word_vector = glove_model['sample']
print("GloVe Vector for 'sample':", glove_word_vector)
```

```
GloVe Vector for 'sample': [-0.31059    0.66222    0.21789   -0.046964  -0.37409    0.71737
  0.92633    0.80566   -0.31682    0.54172   -0.30107   -0.1522
 -0.047403  -0.14923    0.57646    0.76697    0.21433    0.20375
  0.57335   -0.0186    -0.38483   -0.34623    0.10475   -0.48937
 -0.68399   -0.26709    0.70421    0.0032026  0.25875   -0.20735
  0.29176   -0.35039    0.078128   0.69664    0.84063    0.019594
  0.32728    0.49883   -0.4859    -0.59618   -0.13152   -0.58993
 -0.49327   -0.16369   -0.31943    0.13787   -0.75657   -0.98744
```

```
        -0.11669   -0.3329     1.1011     0.40166    0.23021    0.10653
        -0.47182   -1.2705    -0.72376   -0.10318    1.4433     0.011268
         0.29639    0.45875    0.56541    0.55125    1.2032     0.4963
         0.30035   -0.21314    0.36452   -0.07512   -0.18296    0.97547
         0.65208    0.49546    0.18021    0.70638   -0.02527   -0.88635
        -0.31636   -0.34983    0.10328    0.28178   -0.72644   -0.57728
        -1.1255    -0.30205    1.209     -0.23275   -0.34631    0.42445
        -0.48058    0.50546   -0.21942   -0.19309    0.29072    0.26906
        -0.50211   -0.33296   -0.40494    0.29489  ]
```

```
glove_similar_words = glove_model.most_similar('nlp')
print("Words similar to 'nlp' using GloVe:", glove_similar_words)
```

```
Words similar to 'nlp' using GloVe: [('hagelin', 0.6064562797546387), ('oth', 0.5276078581809998), ('grn', 0.5221459865570068),
```

```
class NLPProcessor:
    def _init_(self):
        # Download NLTK data
        nltk.download('punkt')
        nltk.download('stopwords')
        nltk.download('wordnet')
```