text_to_word_sequence

```
keras.preprocessing.text.text_to_word_sequence(text,
                                               filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n
                                               lower=True,
                                               split=" ")
```

文章を単語のリストに分割します.

- **戻り値**: 単語（文字列）のリスト.
- **引数**:
  - **text**: 文字列.
  - **filters**: 句読点などフィルタする文字を含むリスト（あるいはコレクション）. デフォルトは基本的な句読点，タブ，改行を含む'!"#$%&()*+,-./:;<=>?@[\]^_`{|}~\t\n'です.
  - **lower**: 真理値. テキストを小文字にするかどうか.
  - **split**: 文字列. 単語を分割するセパレータ.

one_hot

```
keras.preprocessing.text.one_hot(text,
                                 n,
                                 filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n',
                                 lower=True,
                                 split=" ")
```

文章を単語インデックス（語彙数n）のリストにone-hotエンコードします.

ハッシュ関数として `hash` を使用する `hashing_trick` のラッパーです.

- **戻り値**: [1, n]の整数から構成されるリスト. 各整数は単語をエンコードします（単一性は保証されません）.
- **引数**:
  - **text**: 文字列.
  - **n**: 整数. 語彙数.
  - **filters**: 句読点などフィルタする文字を含むリスト（あるいはコレクション）. デフォルトは基本的な句読点，タブ，改行を含む'!"#$%&()*+,-./:;<=>?@[\]^_`{|}~\t\n'です.
  - **lower**: 真理値. テキストを小文字にするかどうか.
  - **split**: 文字列. 単語を分割するセパレータ.

hashing_trick

```
keras.preprocessing.text.hashing_trick(text,
                                       n,
                                       hash_function=None,
                                       filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n',
                                       lower=True,
                                       split=' ')
```

テキストを固定長のハッシュ空間におけるインデックスの系列に変換します.

- **戻り値**: 単語のインデックスを表す整数のリスト（単一性を保証しない）
- **引数**:
  - **text**: 文字列
  - **n**: ハッシュ空間の次元数
  - **hash_function**: デフォルトはpythonの `hash` 関数で、'md5'か文字列を整数に変換する任意の関数． 'hash'は安定したハッシュ関数ではありません，なので 実行ごとに一貫しません が，'md5'は安定なハッシュ関数です.
  - **filters**: 句読点などフィルタする文字を含むリスト（あるいはコレクション）．デフォルトは基本的な句読点，タブ，改行を含む'!"#$%&()*+,-./:;<=>?@[]^_`{|}~\t\n'です.
  - **lower**: 真理値．テキストを小文字にするかどうか.
  - **split**: 文字列．単語を分割するセパレータ.

Tokenizer

```
keras.preprocessing.text.Tokenizer(num_words=None,
                                   filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n',
                                   lower=True,
                                   split=" ",
                                   char_level=False)
```

テキストをベクトル化する，または/かつ，テキストをシーケンス（= データセット中でランクi（1から始まる）の単語がインデックスiを持つ単語インデックスのリスト）に変換するクラス.

- **引数**: `text_to_word_sequence` と同じです.
  - **num_words**: Noneまたは整数．利用する単語の最大数（もしこの引数が与えられた場合，データセット中の頻度上位num_wordsの単語に制限されます）.
  - **char_level**: Trueなら，全文字はトークンとして扱われる
- **メソッド**:
  - **fit_on_texts(texts)**:
    - **引数**:
      - **texts**: 学習に使う文章のリスト.
  - **texts_to_sequences(texts)**
    - **引数**:
      - **texts**: シーケンスに変換する文章のリスト.
    - **戻り値**: シーケンスのリスト（入力文章ごとに1つ）.

    - **texts_to_sequences_generator(texts)**: 上記のジェネレータ版.
        - **戻り値**: 入力文章ごとに1つのシーケンス.
    - **texts_to_matrix(texts)**:
        - **戻り値**: `(len(texts), num_words)` のshapeをもつNumpy 配列.
        - **引数**:
            - **texts**: ベクトル化する文章のリスト.
            - **mode**: "binary", "count", "tfidf", "freq" のいずれか（デフォルト: "binary"）.
    - **fit_on_sequences(sequences)**:
        - **引数**:
            - **sequences**: 学習に使うシーケンスのリスト.
    - **sequences_to_matrix(sequences)**:
        - **戻り値**: `(len(sequences), num_words)` のshapeをもつNumpy 配列.
        - **引数**:
            - **sequences**: ベクトル化するシーケンスのリスト.
            - **mode**: "binary", "count", "tfidf", "freq" のいずれか（デフォルト: "binary"）.
- **属性**:
    - **word_counts**: 単語（文字列）とそれがfit中に表れた回数をマッピングする辞書. `fit_on_texts` が呼ばれた後にのみセットされます.
    - **word_docs**: 単語（文字列）とfit中に表れた文書/文章の数をマッピングする辞書. `fit_on_texts` が呼ばれた後にのみセットされます.
    - **word_index**: 単語（文字列）とそのランク/インデックス（整数）をマッピングする辞書. `fit_on_texts` が呼ばれた後にのみセットされます.
    - **document_count**: 整数. 訓練に利用された文書（文章／シーケンス）数. `fit_on_texts`, または `fit_on_sequences` が呼ばれた後にのみセットされます.