pad_sequences

```
keras.preprocessing.sequence.pad_sequences(sequences, maxlen=None, dtype='int32')
```

Transform a list of `nb_samples sequences` (lists of scalars) into a 2D Numpy array of shape `(nb_samples, nb_timesteps)`. `nb_timesteps` is either the `maxlen` argument if provided, or the length of the longest sequence otherwise. Sequences that are shorter than `nb_timesteps` are padded with zeros at the end.

- **Return**: 2D Numpy array of shape `(nb_samples, nb_timesteps)`.
- **Arguments**:
    - **sequences**: List of lists of int or float.
    - **maxlen**: None or int. Maximum sequence length, longer sequences are truncated and shorter sequences are padded with zeros at the end.
    - **dtype**: datatype of the Numpy array returned.
    - **padding**: 'pre' or 'post', pad either before or after each sequence.
    - **truncating**: 'pre' or 'post', remove values from sequences larger than maxlen either in the beginning or in the end of the sequence
    - **value**: float, value to pad the sequences to the desired value.

skipgrams

```
keras.preprocessing.sequence.skipgrams(sequence, vocabulary_size,
    window_size=4, negative_samples=1., shuffle=True,
    categorical=False, sampling_table=None)
```

Transforms a sequence of word indexes (list of int) into couples of the form:

- (word, word in the same window), with label 1 (positive samples).
- (word, random word from the vocabulary), with label 0 (negative samples).

Read more about Skipgram in this gnomic paper by Mikolov et al.: Efficient Estimation of Word Representations in Vector Space

- **Return**: tuple `(couples, labels)`.
    - `couples` is a list of 2-elements lists of int: `[word_index, other_word_index]`.
    - `labels` is a list of 0 and 1, where 1 indicates that `other_word_index` was found in the same window

    as `word_index`, and 0 indicates that `other_word_index` was random.
    - if categorical is set to True, the labels are categorical, ie. 1 becomes [0,1], and 0 becomes [1, 0].
- **Arguments**:
    - **sequence**: list of int indexes. If using a sampling_table, the index of a word should be its the rank in the dataset (starting at 1).
    - **vocabulary_size**: int.
    - **window_size**: int. maximum distance between two words in a positive couple.
    - **negative_samples**: float >= 0. 0 for no negative (=random) samples. 1 for same number as positive samples. etc.
    - **shuffle**: boolean. Whether to shuffle the samples.
    - **categorical**: boolean. Whether to make the returned labels categorical.
    - **sampling_table**: Numpy array of shape `(vocabulary_size,)` where `sampling_table[i]` is the probability of sampling the word with index i (assumed to be i-th most common word in the dataset).

---

make_sampling_table

```
keras.preprocessing.sequence.make_sampling_table(size, sampling_factor=1e-5)
```

Used for generating the `sampling_table` argument for `skipgrams`. `sampling_table[i]` is the probability of sampling the word i-th most common word in a dataset (more common words should be sampled less frequently, for balance).

- **Return**: Numpy array of shape `(size,)`.
- **Arguments**:
    - **size**: size of the vocabulary considered.
    - **sampling_factor**: lower values result in a longer probability decay (common words will be sampled less frequently). If set to 1, no subsampling will be performed (all sampling probabilities will be 1).