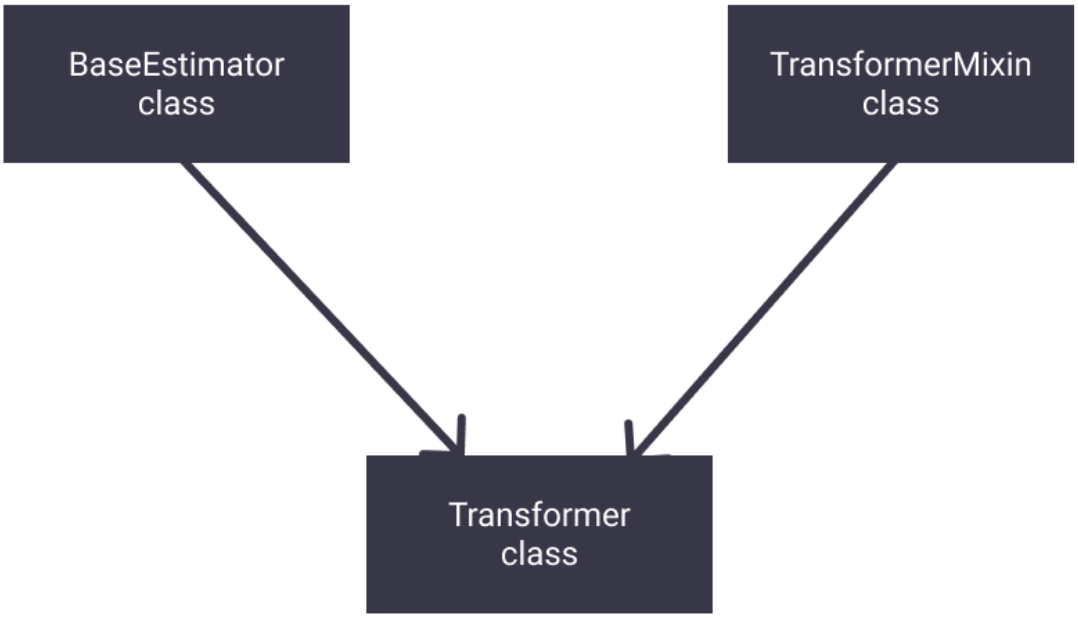
Natural Language Processing
Homework nr. 4

Even though during the last workshop we showed you how to implement all the pre-processing in one single class, usually in NLP pipelines all steps of processing are made by a single class. In today's homework we want to challenge you and invite you to implement the following modules as subclasses of sklearn BaseClasses, making them in such a way all compatible with the sklearn pipeline.

The scikit learn API.

scikit-learn is known as one of the most popular packages for Machine Learning in python. This happened because of its large ecosystem built around the process of building and developing Machine Learning Models. About modeling with python you will talk with Vladimir at the next workshop, but for now we will talk about the transformers ecosystems.

A Transformer in sklearn is a module that takes data, transforms it and returns the changed data. All transformers in sklearn are classes that are subclasses of the **BaseEstimator** and **TransformerMixin** classes.



BaseEstimator class gives every model in sklearn a standard way to be serialized - converted into a binary format and saved into a binary file on a PC. TransformerMixin gives the transformer the ability to use the fit_transform function, which transforms the data directly when passed to the model.

The requirements of a class to become a sklearn compatible transformer.

For a class to become a sklearn compatible transformer it must implement the following 2 functions- fit(), transform(). The fit() function must have the following definition:

- fit(self, X, y=None, **fit_params)

There X represents the feature array that the transformer should learn to transform. y is ignored usually, but is used to make the class compatible with the sklearn API. fit_params represents the additional arguments that you can set, in the function they are present as a dictionary.

The fit function must return the object itself, meaning that you must write return self at the end of the function.

- transform(self, X, y=None, **fit_params)

The transform function takes the same arguments as the fit() function. The difference there is that it returns the transformed data instead of the object itself.

Also the transformers can if you want, implement the fit_transform() function. Usually it is taken from the TransformerMixin because it already implements it, however if you want change you can rewrite it.

It should fit the transformer and return the transformed data.

Task 1: TextNormalizer.

Implement a class using the sklearn API that will remove all the ‘new line’ (\n) and (\r) symbols, will lower all the symbols and keep in the text only the alphabetic symbols, and will return the normalized text.

Task 2: WordsExtractor.

Implement a class using the sklearn API that will remove all the stopwords and hapaxes from the normalized text. The list of stopwords should be passed as a list to the class constructor. The list of hapaxes should be found in the fit function and then used in the transform function.

Task 3: ApplyStemmer.

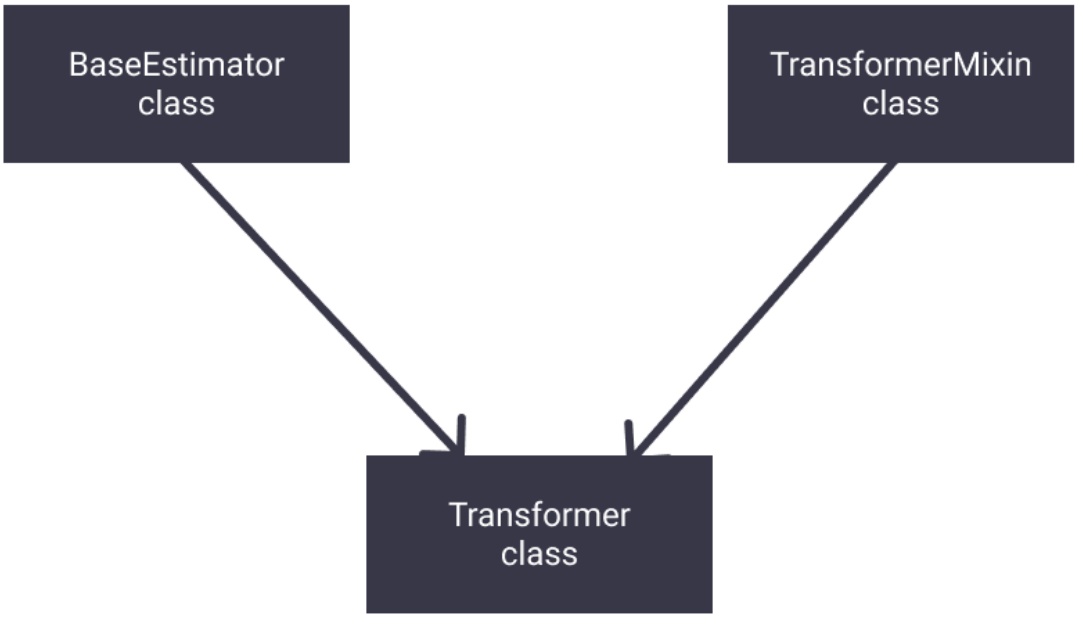
Implement a class using the sklearn API that will take a stammer as a parameter of the constructor. It should apply the stammer on the whole passed corpus and return the cleaned stemmed text.

Chiar și dacă în ultimul workshop noi v-am arătat cum toată preprocesarea textului este realizată într-o singură clasă, de obicei în pipeline-urile de NLP toți pașii de procesare sunt realizați de clase aparte. În această temă de acasă dorim să vă provocăm și vă invităm să implementați următoarele module ca subclase a clasei BaseClass din biblioteca sklearn, făcându-le astfel compatibile cu pipeline-urile din sklearn.

API-ul din sklearn.

scikit-learn este una din cele mai populare pachete pentru Machine Learning în python. Asta s-a întâmplat datorită ecosistemului larg construit în jurul procesului de construire și dezvoltarea a Modelor de Machine Learning. Despre Modelare în python voi veți discuta la următorul workshop cu Vladimir, dar acum noi vom discuta despre ecosistema transformatorilor din sklearn.

Un Transformator în sklearn este un modul care primește ceva date, le transformă și le returnează modificate. Toți transformatorii din sklearn sunt clase care sunt copii are claselor **BaseEstimator** și **TransformerMixin**.



Clasa BaseEstimator oferă fiecărei clase din sklearn un model standardizat de serializare - conversia într-un format binar și salvarea sa într-un fișier binar pe calculator. TransformerMixin oferă transformatorului capacitatea de a folosi funcția fit_transform, care transformă datele direct când sunt transferate modelului.

Cerințele față de o clasă pentru a deveni un transformator compatibil cu sklearn.

Pentru ca o clasă să devină un transformator compatibil cu sklearn ea trebuie să implementeze următoarele funcții: fit() și transform(). Funcția fit() trebuie definita in modul următor:

- fit(self, X, y=None, **fit_params)
  Aici X reprezintă un array cu feature-urile pe care transformatorul trebuie să învețe să le transforme. y este de obicei ignorat, dar este utilizat ca să facă clasa compatibilă cu API-ul sklearn. fit_params reprezintă argumentele adiționale pe care le poți seta. În funcție ele sunt reprezentate ca o ca un dicționar.

  Funcția fit() trebuie să returneze obiectul însuși, însemnând că trebuie să scrii return self la sfârșitul funcției.
- transform(self, X, y=None, **fit_params)

  Funcția transform primește aceleași argument ca și funția fit(). Diferența este că ea returnează date transformate în locul obiectului însuși.

  În plus transformatorii pot dacă dorești să implementeze funcția fit_transform(). Deobicei ea este moștenită de la TransformerMixin deoarece el deja implimeanteă deja această funcție, deși dacă dorești poți să o schimbi.

  Ea trebuie să antreneze transformatorul și să returneze datele transformate.

Task 1. TextNormalizer.

Implementează o clasă utilizând API-ul sklearn care va șterge toate simbolurile ‘new line’ (\n) și (\r), va transforma toate simbolurile în minuscule menținând doar simbolurile alfabetice și va returna textul normalizat.

Task 2. WordsExtractor.

Implementează o clasă utilizând API-ul sklearn care va șterge toate cuvintele stop (stopwords) și hapaxele din textul normalizat. Lista de civinte stop trebuie transmisă ca o listă constructorului clasei. Lista hapaxelor trebuie găsită în funcția fit() și folosită apoi în funcția transform.

Task 3: ApplyStemmer.

Implementează o clasă utilizând API-ul sklearn care va primi un stemmer ca parametru al constructorului. Aceasta trebuie să aplice stemmer-ul pe tot corpus-ul și să returneze textul curat.