Main module

```
class TextVisualizer
    def __init__(self, path='docs\\'):
        docs = self.docReader(path)
        self.textTokens = self.tokenizer(docs)
        self.sentences = self.sentenceizer(docs)
```

Die Hauptklasse **TextVisualizer** ruft bei Erstellung eines Objekts die Methoden **docReder**, **tokenizer** und **sentenceizer** auf, um die Verarbeitung der Korpusdokumente anzustoßen. **Parameter**

- `path` bestimmt die Position der Textdokumente relativ zum Pfad der Klasse. Standard: `docs\\`

tokenizer

```
tokenizer(self, documents)
```

Der bei Erstellung eines Objekts aufgerufene **tokenizer** öffnet jedes Dokument mit *utf-8* encoding und tokenisiert es.

Returnvalue: Geschachtelte Liste mit jedem tokenisierten Dokument

sentenceizer

```
sentenceizer(self, documents)
```

Der bei Erstellung eines Objekts aufgerufene **sentenceizer** öffnet jedes Dokument mit *utf-8* encoding und spaltet es an Satzgrenzen.

Returnvalue: Geschachtelte Liste mit allen Sätzen eines Dokuments

main

```
main(self)
```

Die **main** Methode kann aufgerufen werden, um alle notwendigen Standardmethoden(*createDictionary, createBow, createLsiModel, prepLsiData, docContrLsi, detAvgSentLength, detAvgWordLength, mostCommonTypes, showMainPlots*) auszuführen und den plot anzustoßen. Resultat sind Grafiken mit den Features:

- Most common Types in corpus
- Avarage Token/Sentence Length in corpus
- LSI with keywords by topic and document contribution

createDictionary

```
createDictionary(self, textTokens, excludeMostCommon=True, excludeTypes=[])
```

Die Methode erstellt mit Hilfe von *Gensim* ein Dictionary (mapping zwischen normalisierten Worten und IDs) für den Corpus. **Parameter**

- `textTokens` ist der Rückgabewert der Methode **tokenizer**
- `excludeMostCommon` gibt an, ob einige der häufigsten Wörter der englischen Sprache von dem Prozess ausgeschlossen werden sollen, falls sie in den Dokumenten vorkommen. Standard: `True`
- `excludeTypes` ist eine Liste, die individuell befüllt werden kann, um bestimmte Types vom Prozess auszuschließen. Standard: `Null`

Returnvalue: Dictionary des Korpus

createBow

```
createBow(self, dictionary, textTokens)
```

Die Methode erstellt mit Hilfe von *Gensim* ein bag-of-words (mapping zwischen Wort IDs und Frequenz) für den Corpus. **Parameter**

- `textTokens` ist der Rückgabewert der Methode **tokenizer**
- `dictionary` ist der Rückgabewert der Methode **createDictionary**

Returnvalue: Liste von Tuplen(wort_ID, wort_frequenz)

createLsiModel

```
createLsiModel(self, bow, dictionary, df=False)
```

Die Methode erstellt mit Hilfe von *Gensim* ein lsi-model für den Corpus. **Parameter**

- `bow` ist der Rückgabewert der Methode **createBow**
- `dictionary` ist der Rückgabewert der Methode **createDictionary**
- `df` gibt an, ob das bow-model vorher in ein tfidf-model umgewandelt werden soll. Standard: `False`

Returnvalue: Ein wrapped lsi-model des Korpus mit 3 topics

docContrLsi

```
docContrLsi(self, lsi)
```

Die Methode berechnet den Anteil, den die Korpusdokumente zu den lsi-topics beitragen.

Parameter

- `lsi` ist ein Vektor erstellt mit dem lsimodel aus dem bow z. B. `lsimodel[bow]`

Returnvalue: Geschachtelte Liste mit den Werten geordnet nach topic

detAvgSentLength

```
detAvgSentLength(self, sentences)
```

Die Methode ermittelt die durschnittliche Satzlänge im Korpus **Parameter**

- `sentences` ist der Rückgabewert der Methode **sentenceizer**

Returnvalue: Tupel(Satzlänge in Zeichen, Satzlänge in Worten)

detAvgWordLength

```
detAvgWordLength(self, textTokens)
```

Die Methode ermittelt die durschnittliche Wortlänge im Korpus **Parameter**

- `textTokens` ist der Rückgabewert der Methode **tokeneizer**

Returnvalue: Wortlänge (Int)

mostCommonTypes

```
mostCommonTypes(self, dictionary, bow, excludeTypes=[], n=30)
```

Die Methode ermittelt die häufigsten Wörter im Korpus **Parameter**

- `dictionary` ist der Rückgabewert der Methode **createDictionary**
- `bow` ist der Rückgabewert der Methode **createBow**
- `excludeTypes` ist eine Liste, die individuell befüllt werden kann, um bestimmte Types vom Prozess auszuschließen. Standard: `Null`
- `n` gibt die Anzahl der zu ermittelnden Wörter an. Standard: `30`

Returnvalue: Tupel(Liste[Wörter], Liste[Frequenz])

prepLsiData

```
prepLsiData(self, lsimodel)
```

Die Methode bereitet die topics des lsi-models zu Darstellung vor **Parameter**

- `lsimodel` ist der Rückgabewert der Methode **createLsiModel**

Returnvalue: Geschachtelte Liste [[topic1], [topic2], [topic3]]

showMainPlots

```
showMainPlots(self, data_MCT, data_avg, data_lsi, data_contr)
```

Die Methode bereitet alle gewonnen Daten zu Darstellung vor und zeigt den Plot. **Parameter**

- `data_MCT` ist der Rückgabewert der Methode **mostCommonTypes**
- `data_avg` ist der Rückgabewert der Methoden **avgSentLength + avgWordLength**
  - `data_lsi` ist der Rückgabewert der Methode **prepLsiData**
  - `data_contr` ist der Rückgabewert der Methode **docContrLsi**

Other features

similarityReq

```
similarityReq(self, document='compare\\')
```

Die Methode macht mit Hilfe von *Gensim* einen Ähnlichkeitsvergelich zwischen den Korpusdokumenten und einem zusätzlichen Dokument. **Parameter**

- `document` gibt die Position des zusätzlichen Dokuments relativ zum Projektordner an. Standard: `compare\\`

Returnvalue: Tupel(Liste[Korpusdokumentnummer], Liste[Ähnlichkeit in Prozent])

showSimPlot

```
showSimPlot(self, data_sims)
```

Plottet die Daten eines similarity requests. **Parameter**

- `data_sim` ist der Rückgabewert der Methode **similarityReq**

wordHistory

```
wordHistory(self, typ, granularity)
```

Die Methode zeigt den "zeitlichen" Verlauf eines Wortes in den Korpusdokumenten auf. **Parameter**

- `typ` bestimmt das aufzuzeichnende Wort

- `granularity` bestimmt wie viele Wörter eines Dokuments zu einem x-Achsenabschnitt zusammengefasst werden sollen

Returnvalue: Tupel(Liste[Abschnitt], Liste[Wortvorkommen], Wort)

showHistPlot

```
showHistPlot(self, data_hist)
```

Plottet die Daten einer word history. **Parameter**

- `data_hist` ist der Rückgabewert der Methode **wordHistory**

Plot

Das Plot-Modul arbeitet im Hintergund und allein mit *Bokeh*-Funktionen und ist das Rückgrat des TextViz. Es werden Bar-, Pie- und Linecharts verwendet, um den jeweligen Aufgaben visuell gerecht zu werden.