Data Mining & Pattern Recognition

Versuch 5:

Merkmalsextraktion mit der Nicht-Negativen Matrixfaktorisierung

Datum:	
Namen:	
Unterschrift Betreuer:	

Prof. Dr. Johannes Maucher
Medieninformatik Bachelor (MIB)
WS 12/13

Inhaltsverzeichnis

1 Einführung

1.1 Lernziele

In diesem Versuch sollen Kenntnisse in folgenden Themen vermittelt werden:

- **RSS Feeds:** Struktur von RSS Feeds analysieren und parsen mit dem *Universal Feed Parser*.
- **Dokument Analyse:** Die Häufigkeit aller Worte in einem Dokument (Inhalt des RSS Feeds) zählen und in einem Array verwalten.
- **Merkmalsextraktion:** Bestimmung von Merkmalen (hier auch: **Topics**)[1] mit der *Non Negative Matrix Factorization*.
- **Zuordnung**: Wie setzen sich die Topics aus den Wörtern zusammen? Wie stark sind die gefundenen Topics in den Artikeln vertreten?
- **Dokument Clustering:** Mit der NNMF kann auch ein Clustering realisiert werden. Jeder Topic repräsentiert ein Cluster. Jedes Dokument wird dem Cluster zugeordnet, dessen Topic am stärksten in ihm vertreten ist.

Sämtliche Verfahren und Algorithmen werden in Python implementiert.

1.2 Protokoll

Abzugeben sind pro Gruppe spätestens vor Beginn des nächsten Versuchs:

- Antworten auf alle in dieser Anleitung gestellten Fragen. Die Antworten sollten in einem elektronischen Dokument verfasst und als .pdf bereitgestellt werden. Die Antworten sind entsprechend der Kapitelnummerierung in dieser Anleitung zu ordnen.
- Alle Dateien von denen in dieser Anleitung ein Abspeichern gefordert wird.

Die Durchführung und Dokumentation eigener Experimente, die über die Aufgabenstellungen in diesem Dokument hinausgehen, wird ausdrücklich unterstützt und in der Bewertung entsprechend berücksichtigt.

1.3 Theorie zur Vorbereitung

Stellen Sie sich vor Sie möchten in eine eigene Webseite die RSS Feeds einer Menge von Nachrichtenservern einbinden. Da die unterschiedlichen Server wahrscheinlich Artikel zu den gleichen Themen anbieten, werden die Inhalte einiger Artikel ähnlich sein. Mit der *Nicht Negativen Matrixfaktorisierung (NNMF)* kann für eine große Menge von Dokumenten eine Menge von Themen (Topics) ermittelt werden, auf die sich die Dokumente beziehen. Damit ist es u.a. möglich

- die Dokumente thematisch zu ordnen
- zu jedem Thema nur ein Dokument anzuzeigen

[1] Allgemein spricht man von Merkmalen. Im Fall, dass die NNMF auf Dokumente angewandt wird werden die Merkmale auch mit *Topics* oder *Themen* bezeichnet

1.3.1 Ähnlichkeiten bestimmen und relevante Merkmale extrahieren

Eine Sammlung von Dokumenten - in diesem Versuch die Menge aller Nachrichten der angegebenen Feeds - kann in einer Artikel/Wort-Matrix repräsentiert werden. Jede Zeile dieser Matrix gehört zu einem Dokument. Für jedes Wort, das mindestens in einem der Dokumente vorkommt, ist eine Spalte vorgesehen. Der Matrixelement in Zeile i, Spalte j beschreibt wie häufig das Wort in Spalte j im zur Zeile i gehörenden Dokument vorkommt.

Unter der Annahme, dass Artikel umso ähnlicher sind, je mehr Worte in diesen gemeinsam vorkommen, kann auf der Grundlage dieser Matrix die Ähnlichkeit zwischen den Artikeln berechnet werden. Hierzu könnte die Matrix z.B. einfach einem *Hierarchischen Clustering* übergeben werden. Das hierarchische Clustering weist jedoch im Fall einer großen Menge von zu vergleichenden Objekten zwei wesentliche Nachteile auf: Erstens ist die wiederholte Berechnung der Distanzen zwischen allen Artikeln/Clustern extrem rechenaufwendig, zweitens ist die Darstellung einer großen Anzahl von Objekten im Dendrogramm nicht mehr übersichtlich.

Für das Auffinden von Assoziationen zwischen Dokumenten hat sich in den letzten Jahren die Methode der nicht-negativen Matrix Faktorisierung (nnMF) etabliert. Mit dieser Methode kann eine Menge von wesentlichen Merkmalen berechnet werden, anhand derer sich die Dokumente clustern lassen, d.h. Dokumente des gleichen Clusters repräsentieren das gleiche Merkmal (Thema). Ein solches Merkmal wird durch eine Menge von Worten beschrieben, z.B. {*Obama, visit, Germany* } oder {*crisis, bank, credit* }. Neben der Merkmalsextraktion stellt die relativ geringe Komplexität einen weiteren Vorteil der nnMF dar. Durch die Darstellung der Artikel/Wort-Matrix als Produkt von 2 Faktormatrizen müssen deutlich weniger Einträge gespeichert werden

1.3.2 Nicht-Negative Matrix Faktorisierung: Die Idee

Die Artikel/Wort-Matrix wird im Folgenden mit A bezeichnet. Sie besitzt r Zeilen und c Spalten, wobei r die Anzahl der Artikel und c die Anzahl der relevanten Worte in der Menge aller Artikel ist. Durch Multiplikation der Matrix A mit dem Vektor v (`wordvec`: Vektor der alle relevanten Worte enthält) werden die Worte den Artikeln a (`articletitles`: Vektor der alle Artikeltitel enthält) zugeordnet:

$$a = A * v. \tag{1}$$

Die Idee der nnMF besteht darin die Matrix A als Produkt zweier Matrizen W und H darzustellen,

$$A = W * H \tag{2}$$

wobei alle Elemente in W und H größer oder gleich Null sein müssen. Die Matrixmultiplikation erfordert, dass die Anzahl der Zeilen m in H gleich der Anzahl der Spalten in W sein muss. Durch die Faktorisierung der Matrix A wird die Zuordnung der Wörter des Wortvektors v zu den Artikeln des Vektors a in zwei Stufen zerlegt.

$$f = H * v \tag{3}$$
$$a = W * f \tag{4}$$

In der ersten Stufe werden durch die Multiplikation von v mit der Matrix H die Wörter einem sogenannten Merkmalsvektor f mit m Elementen zugewiesen. In der zweiten Stufe werden durch die Multiplikation des Merkmalsvektor f mit der Matrix W die einzelnen Merkmale den Artikeln in a zugeordnet. Die Matrix H definiert also aus welchen Wörtern die Merkmale gebildet werden. Sie wird deshalb **Merkmalsmatrix** genannt. Die Matrix W hingegen beschreibt mit welchem Gewicht die

einzelnen Merkmale in den verschiedenen Artikeln auftreten. Sie wird deshalb **Gewichtungsmatrix** genannt.

Daraus folgt: Wenn eine Faktorisierung der Matrix A gefunden wird, dann werden damit auch relevante Merkmale, also die Themen, definiert, hinsichtlich derer die Artikel effizient kategorisiert werden. Durch die Matrixfaktorisierung wird eine **Merkmalsextraktion** realisiert.

1.3.3 Berechnung der Matrix Faktoren

Für die Berechnung der Faktoren wurde in [LEE] eine iterative Methode vorgestellt, die derzeit wohl am häufigsten angewandt wird und auch in dieser Übung implementiert werden soll. Der Algorithmus besteht aus folgenden Schritten:

1. Gebe die zu faktorisierende Matrix A ein. r sei die Anzahl der Zeilen und c die Anzahl der Spalten von A.
2. Wähle die Anzahl m der Merkmale, mit $m < c$.
3. Lege eine $m \times c$ Matrix H an mit initial zufälligen Elementen
4. Lege eine $r \times m$ Matrix W an mit initial zufälligen Elementen
5. Wiederhole bis maximale Anzahl der Iteration erreicht oder Kosten k unter vordefinierter Schwelle:
   (a) Berechne aktuelles Produkt $B = W * H$ und bereche die Kostenfunktion
   $$k = \|A - B\|^2 = \sum_{i,j} (A_{i,j} - B_{i,j})^2 \tag{5}$$
   (b) Anpassung der Matrix H durch folgende Neuberechnung der Matrixelemente
   $$H_{i,j} := H_{i,j} \frac{(W^T * A)_{i,j}}{(W^T * W * H)_{i,j}} \tag{6}$$
   (c) Anpassung der Matrix W durch folgende Neuberechnung der Matrixelemente
   $$W_{l,i} := W_{l,i} \frac{(A * H^T)_{l,i}}{(W * H * H^T)_{l,i}} \tag{7}$$

In [LEE] ist bewiesen, dass durch die Anpassungsroutinen (6) und (7) die Kosten k (5) monoton abnehmen und in einem Minimum konvergieren. Der Algorithmus ist jedoch nicht optimal weil das gefundene Minimum ein lokales Minimum sein kann.

1.3.4 Vor dem Versuch zu klärende Fragen

- Was versteht man unter Artikel/Wort-Matrix? Wie sieht diese im aktuellen Versuch aus?
- Wie multipliziert man die Matrix
$$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \end{pmatrix}$$

mit dem Vektor

$$v = \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

- Was versteht man im Kontext der NNMF unter
  - Merkmalsmatrix
  - Gewichtsmatrix
- Wie definieren die Zeilen der Merkmalsmatrix die einzelnen Merkmale (Topics)?
- Was definieren die Zeilen der Gewichtungsmatrix?
- Wie werden in Numpy zwei Arrays (Typ numpy.array)
  - im Sinne der Matrixmultiplikation miteinander multipliziert?
  - elementweise multipliziert?
- Wie wird die Transponierte eines Numpy-Arrays berechnet?

2 Durchführung

Schreiben Sie alle im folgenden beschriebenen Funktionen in eine Python Datei

newsfeatures.py

und das Programm zum Test der Funktionen in die Datei

newstest.py

Achten Sie darauf, dass Sie den Quellcode gut dokumentieren.

2.1 RSS Nachrichten Feeds einbinden und parsen

1. Legen Sie zunächst eine Liste mit etwa 10 verschiedenen Web-Adressen von RSS Nachrichten Feeds an. Wenn Sie nicht selbst nach Feeds suchen möchten, können Sie die folgenden verwenden:

Listing 1: Liste von RSS Newsfeed Adressen

```
feedlist=['http://feeds.reuters.com/reuters/topNews',
      'http://feeds.reuters.com/reuters/businessNews',
      'http://feeds.reuters.com/reuters/worldNews',
      'http://feeds2.feedburner.com/time/world',
      'http://feeds2.feedburner.com/time/business',
      'http://feeds2.feedburner.com/time/politics',
      'http://rss.cnn.com/rss/edition.rss',
      'http://rss.cnn.com/rss/edition_world.rss',
      'http://newsrss.bbc.co.uk/rss/newsonline_world_edition/business/rss.xml',
      'http://newsrss.bbc.co.uk/rss/newsonline_world_edition/europe/rss.xml',
      'http://www.nytimes.com/services/xml/rss/nyt/World.xml'
      'http://www.nytimes.com/services/xml/rss/nyt/Economy.xml'
      ]
```

Tipp: Durch die Verwendung englischsprachiger Feeds können Sie spezielle Anpassungen für den Umgang mit Umlauten vermeiden. Es ist unbedingt darauf zu achten, dass eine große Anzahl von Nachrichten geladen wird - je mehr umso besser.

2. Binden Sie die Bibliothek *feedparser* ein. Übergeben Sie der Funktion `feedparser.parse()` die Elemente der angelegten Feed-Liste.
   - Was für eine Datenstruktur liefert die Funktion zurück?
   - Wie kann auf den Titel und die Beschreibung des RSS-Feeds zugegriffen werden?
   - Lassen Sie sich die Titel und Inhalte der aktuellen Artikel der von Ihnen ausgewählten Nachrichten-Feeds anzeigen.

2.2 Sammeln und speichern aller Worte der aktuellen Artikel aller eingebundenen Feeds

2.2.1 Worte sammeln

Schreiben Sie eine Funktion `getarticlewords()` mit

Übergabeparameter: keine

Rückgabewerte:
- `allwords` ist ein Dictionary dessen Keys die Worte aller gesammelten Artikel sind. Der zu jedem Key gehörende Wert ist die Anzahl wie oft das Wort insgesamt vorkommt.
- `articlewords` ist eine Liste mit sovielen Elementen wie Artikel in der Sammlung sind. Jedes Listenelement ist ein Dictionary, welches die Worte des jeweiligen Artikels als Key enthält und als Wert die Worthäufigkeit.
- `articletitles` ist eine Liste mit so vielen Elementen wie Artikel in der Sammlung sind. Jedes Element ist der Artikeltitel als String.

Die Struktur der Rückgabewerte ist exemplarisch in Listing 2 dargestellt.

Innerhalb der Funktion sollen

1. die 3 Rückgabe-Datenstrukturen initialisiert werden.
2. für jeden Feed und für jeden Artikel innerhalb eines Feeds
   (a) der `title` und die `description` des Artikels zu einem String zusammengefügt werden. Die `description` enthält i.A. noch HTML-Markup. Diese Markup soll vor dem Zusammenfügen im neuen String entfernt werden. Das Herausfiltern des Markup kann mit der Funktion `stripHtml(h)` realisiert werden (siehe Listing 3).
   (b) Der im vorigen Schritt gebildete String aus Artikeltitel und Markup-freiem Artikeltext soll nun in seine einzelnen Wörter zerlegt werden. Diese Zerlegung kann mit der Funktion `separatewords(text)` realisiert werden (siehe Listing 3). In der Funktion werden Wörter, die keine Information tragen (Stopwörter) herausgefiltert.
   (c) Die Inhalte der von `separatewords(text)` zurückgegebenen Datenstruktur werden in die Datenstrukturen `allwords` und `articlewords` aufgenommen. Der Titel des aktuellen Artikels wird an die Liste `articletitles` angehängt.

Listing 2: Exemplarischer Auszug der Rückgabewerte der Funktion `getarticlewords()`

```
allwords=
{'aided': 1, 'actionfor': 1, 'ambitionsthe': 1, 'reboot': 1, 'four': 2, ...
articlewords[0]=
{'accuses': 1, 'into': 1, 'racism': 1, 'racial': 2, 'debate': 1,...
articletitles=
[u'Obama stokes racial passions, police anger', u'Mayors, rabbis arrested in NJ',...
```

Listing 3: Die Funktionen `stripHTML()` und `separatewords()`

```
def stripHTML(h):
  p=''
  s=0
  for c in h:
    if c=='<': s=1
    elif c=='>':
      s=0
      p+=' '
    elif s==0: p+=c
  return p

from nltk.corpus import stopwords
sw=stopwords.words('english')
def separatewords(text):
  splitter=re.compile('\\W*')
  return [s.lower() for s in splitter.split(text) if len(s)>4 and s not in sw]
```

- Erklären Sie den Ablauf und die Rückgabewerte der Funktionen `stripHtml(h)` und `separatewords(text)` und nehmen Sie diese in das File *newsfeatures.py* auf.
- Implementieren Sie die Funktion `getarticlewords()` im File *newsfeatures.py*.

2.2.2 Die Artikel/Wort-Matrix

Schreiben Sie ein Funktion `makematrix(allw,articlew)` mit

Übergabeparameter: die von `getarticlewords()` zurückgegebenen Datenstrukturen `allwords` und `articlewords`

Rückgabewerte:

- eine Liste `wordvec`, die alle Worte aus `allwords` enthält, ausgenommen der Worte, die
  - insgesamt weniger als 4 mal vorkommen, oder
  - in mehr als 60 % aller Artikel vorkommen
- die **Artikel/Wort-Matrix** als zweidimensionale Liste `wordInArt`,
  - deren erste Dimension (die Zeilen) die einzelnen Artikel repräsentieren
  - deren zweite Dimension (Spalten) die Worte in `wordvec` repräsentieren

  D.h. wenn Zeile i den i.ten Artikel repräsentiert und Spalte j das j.te Wort innerhalb von `wordvec`, dann gibt das Element `wordInArt[i][j]` in Zeile i, Spalte j, an wie häufig das j.te Wort im i.ten Artikel auftritt.

- Implementieren Sie die Funktion `makematrix(allw,articlew)` im File *newsfeatures.py*.
- Schreiben Sie die von `makematrix(allw,articlew)` zurückgegebene Matrix in ein Textfile. Die Elemente innerhalb einer Zeile sollten durch Kommas getrennt werden. In die erste Zeile des Textfiles sollten die Worte aus `wordvec` geschrieben werden.

2.2.3 Repräsentation der Artikel/Wort-Matrix als Numpy-Matrix

Da im Algorithmus der *nnMF* eine Vielzahl von Matrix-Operationen durchzuführen sind, bietet sich für die Implementierung die Anwendung von Funktionen aus der Numpy-Bibliothek an. Hierzu muss auch die oben bereits erstellte Artikel/Wort-Matrix als Numpy-Matrix dargestellt werden. Dies läßt sich einfach durch `A=numpy.matrix(wordmatrix)` realisieren. Bevor diese Matrix der Funktion für die nnMF übergeben wird, muss sichergestellt werden, dass sie keine All-Nullen Zeilen enthält. Eine entsprechende Abfrage und Filterung kann entweder bereits beim Erstellen der Artikel/Wort-Matrix (im vorigen Abschnitt) oder nach der Umwandlung in eine Numpy-Matrix durchgeführt werden. Dabei ist darauf zu achten, dass das Löschen einer Zeile in der Artikel/Wort-Matrix auch das Löschen des entsprechenden Artikeltitels in der Liste `articletitles` erfordert.

2.2.4 Implementierung der NNMF

Die Implementierung der NNMF ist entsprechend der im Theoriekapitel (1.3) durchzuführen.

- Implementieren Sie die Funktion `cost(A,B)` im File *newsfeatures.py*. Dieser Funktion werden zwei Numpy-Matrizen A und B übergeben. Zurück geliefert werden die nach Formel (5) berechneten Kosten. Diese Funktion wird von der im folgenden beschriebenen Funktion `nnmf(A,m,it)` benutzt.
- Implementieren Sie die Funktion `nnmf(A,m,it)` im File *newsfeatures.py*. In dieser Funktion soll der oben beschriebene Algorithmus für die Nicht-negative Matrix Faktorisierung ausgeführt werden. Der Funktion wird die zu faktorisierende Matrix A, die Anzahl der Merkmale m und die Anzahl der Iterationen it übergeben. Die Funktion gibt die gefundenen Faktoren W und H zurück. In jeder Iteration sollen mit der Funktion `cost(A,B)` die Kosten berechnet werden. Sobald diese kleiner als 5 sind soll der Algorithmus mit der Rückgabe der Faktoren W und H terminieren.

Tipp für die Implementierung elementweiser Operationen von Matrizen: Für elementweise Operationen müssen in Python/Numpy nicht alle Elemente über Schleifen explizit berechnet werden. Eine elementweise Anpassung aller Matrixelemente kann kompakt programmiert werden indem die beteiligten Matrizen für diese Operationen als Arrays implementiert werden. Sollen z.B. die zwei gleich großen Matrizen U und V elementweise multipliziert werden, dann wäre der entsprechende Programmcode `array(U)*array(V)`

2.3 Anzeige der Merkmale und der Gewichte

Im vorigen Abschnitt wurde die Merkmalsmatrix H und die Gewichtsmatrix W berechnet. Diese Matrizen können natürlich am Bildschirm ausgegeben werden, was jedoch nicht besonders informativ ist. Aus den Matrizen können jedoch die Antworten für die folgenden interessanten Fragen berechnet werden:

- In welchen Artikeln sind welche Merkmale stark vertreten?
- Wie lassen sich die insgesamt m Merkmale beschreiben, so dass aus dieser Merkmalsbeschreibung klar wird, welches Thema den Artikeln, in denen das Merkmal stark vertreten ist, behandelt wird?

Die Antwort auf die erste Frage ergibt sich aus der Gewichtsmatrix W. Für die Beantwortung der zweiten Frage wird die Merkmalsmatrix H herangezogen.

2.3.1 Beschreibung der Merkmale

Die Merkmalsmatrix H beschreibt, wie stark die Worte aus `wordvec` in jedem Merkmal enthalten sind. Jede Zeile von H gehört zu einem Merkmal, jede Spalte von H gehört zu einem Wort in `wordvec`.
Es bietet sich an jedes Merkmal einfach durch die $N = 6$ Wörter aus `wordvec` zu beschreiben, welche am

stärksten in diesem Merkmal enthalten sind. Hierzu muss für jedes Merkmal die entsprechende Zeile in H nach den $N = 6$ größten Werten durchsucht bzw. geordnet werden. Die entsprechenden Spalten dieser Matrixelemente verweisen dann auf die $N = 6$ wichtigsten Worte des Merkmals.
Tipp für die Implementierung: Legen Sie für jedes Merkmal i eine Liste an. Die Listenlänge ist durch die Anzahl der Worte in `wordvec` (d.h. die Anzahl der Spalten in H) gegeben. Jedes Listenelement j enthält selbst wieder 2 Elemente: An erster Stelle den entsprechenden Wert $H_{i,j}$ der Merkmalsmatrix, an der zweiten Stelle das j.-te Wort in `wordvec`. Nachdem die Liste angelegt ist, kann sie mit `listname.sort()` in aufsteigender Reihenfolge sortiert werden. Die abnehmende Sortierung erhält man mit `listname.sort().reverse()`. Danach geben die $N = 6$ ersten Listenelemente die für das Merkmal i wichtigsten Worte an.

2.3.2 Präsenz der Merkmale in den Artikeln

Die Gewichtsmatrix W beschreibt, wie stark die m Merkmale in den Artikeln aus `articletitles` enthalten sind. Jede Zeile von W gehört zu einem Artikel, jede Spalte von W gehört zu einem Merkmal.
Die Berechnung der $M = 3$ gewichtigsten Merkmale für jeden Artikel in `articletitles` kann analog zu der oben beschriebenen Berechnung der $N = 6$ wichtigsten Worte eines Merkmals berechnet werden.

2.3.3 Implementierung

Implementieren Sie eine Funktion `showfeatures(w,h,titles,wordvec)`, welche wie oben beschrieben

- für jedes Merkmal die $N = 6$ wichtigsten Worte am Bildschirm ausgibt
- für jeden Artikel die $M = 3$ wichtigsten Merkmale am Bildschirm ausgibt

Übergabeparameter der Funktion sind die Merkmalsmatrix H, die Gewichtungsmatrix W, die Liste aller Artikeltitel `articletitles` und die Liste aller Worte `wordvec`.

Geben Sie mindestens 3 Merkmale an und zu jedem Merkmal mindestens 3 Artikel, die das jeweilige Merkmal behandeln.

Literatur

[FP] Universal Feed Parser; `http://feedparser.org`

[TS] Toby Segaran; *Kollektive Intelligenz*; O'Reilly Verlag; 2008

[LEE] D.D. Lee, H.S. Seung; *Algorithms for Non-negative Matrix Factorization*; `http://hebb.mit.edu/people/seung/papers/nmfconverge.pdf`