

Verse des Alten Testaments clustern

Belegarbeit

eingereicht am Fachbereich

Informatik

der Hochschule Zittau/Görlitz (HAW)

als Prüfungsleistung im Fach

Data Mining

vorgelegt von:

Christof Ochmann (35989)

Ingo Körner (40586)

Görlitz, 11. Juli 2012

Betreuer: Prof. ten Hagen

Abstract

Inhaltsverzeichnis

Literaturverzeichnis	VI
1 Theorie	1
1.1 Einleitung	1
1.2 Aufgabenstellung	1
1.3 Relevanz des Forschungsgegenstandes	1
1.4 Der aktuelle Wissensstand	2
1.5 Testrechner	2
1.6 Der Aufbau des Alten Testaments	2
1.7 ARFF	3
1.8 Wie kann das AT in das ARFF-Format überführt werden?	5
1.9 Analyse Text2ARFFConverter	5
1.10 Entwurf Text2ARFFConverter	6
1.11 Nach welchen Wörter sollte geclustert werden?	8
1.12 Hürden beim Einlesen der ARFF-Datei	8
1.13 SimpleKMeans	9
1.14 Weitere Cluster-Algorithmen	10
1.15 geclusterte Instanzen wieder in Verse verwandeln	10
2 Umsetzung	13
2.1 Zusammenfassung	13
2.2 Ausblick	13
A Codebeispiele	14
B Arbeitsaufteilung	18
C Eigenständigkeitserklärung	19

Abbildungsverzeichnis

1.1	Aufbau des AT	3
1.2	Analyseklassendiagramm Text2ARFFConverter	6
1.3	Entwurfsklassendiagramm Text2ARFFConverter	7
1.4	Ausführungszeiten von SimpleKMeans	9
1.5	Analyseklassendiagramm Text2ClusterFile	11
1.6	Entwurfsklassendiagramm Text2ClusterFile	12

Listings

A.1	alle Tabellen erstellen	14
A.2	Datenimport über COPY	15
A.3	Primär- und Fremdschlüssel hinzufügen	16
A.4	Indexe auf Spalten legen	17

Abkürzungsverzeichnis

DBMS	Datenbankmanagementsystem
ERD	Entity-Relationship Diagram
OLAP	Online Analytical Processing
SQL	Structured Query Language

Literaturverzeichnis

- [1] Martin, Robert C. (2008): Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall International
- [2] Freeman, Eric (2007): Entwurfsmuster von Kopf bis Fuß. O'REILLY
- [3] Peter Eisentraut, Bernd Helmle. (2009): PostgreSQL-Administration. O'REILLY
- [4] <http://www.cs.waikato.ac.nz/ml/weka/arff.html> (08.06.2012)
- [5] <http://wiki.pentaho.com/display/DATAMINING> (08.06.2012)

1 Theorie

1.1 Einleitung

Ziel dieses Projektes ist es, ähnliche Verse im Alten Testament, kurz AT, zu finden und zu gruppieren, d.h. zu clustern. Dazu gibt es verschiedene Clustering-Algorithmen. Diese Algorithmen bzw. die von ihnen erzeugten Cluster können miteinander verglichen werden.

1.2 Aufgabenstellung

Durch Clustering werden Ähnlichkeiten in großen Datenbeständen gefunden. In diesem Projekt werden mit Hilfe von Clustering-Algorithmen Verse des AT geclustert. Dabei sollen einander ähnliche Verse in einem Clustern zusammenfasst werden, d.h. Datensätze, die sich ähneln, kommen in dasselbe Cluster. In diesem Projekt werden verschiedene Clusteralgorithmen auf das Alte Testament angewendet. In Werkzeugen wie Weka oder ELKI sind diese Algorithmen bereits implementiert und können genutzt werden. Es soll nur auf einer Maschine geclustert werden, d.h. skalierbares Datamining mit Apache Mahout wird in dieser Arbeit nicht behandelt.

1.3 Relevanz des Forschungsgegenstandes

Der Forschungsgegenstand dieser Arbeit ist, die Verse des AT mit verschiedenen Clusteralgorithmen auf einem handelsüblichen Laptop zu clustern. Dafür wird aus einer Menge möglicher Clustersoftware eine passende ausgewählt. Der Forschungs-

gegenstand ist relevant, da bisher noch keine Ergebnisse für das Clustern von Versen des AT mit dem Testrechner vorliegen. Ziel der Forschung ist es, geeignete Clusteralgorithmen zu finden und mit verschiedenen Parametern auszuführen - wie Anzahl Cluster bzw. Anzahl Attribute nach denen geclustert werden soll. Es wird sich vertiefend in eine Clustersoftware und den Clusteralgorithmen eingearbeitet. Das geschieht z.B. unter Zuhilfenahme von Büchern und Online-Ressourcen. In diesen Medien ist der Forschungsstand zu Software und Algorithmen dokumentiert. Bei der Erstellung der passenden Eingabeformate und zur Auswertung der geclusterten Ergebnisse sind zudem Programme zu erstellen, bei denen technische Probleme gelöst werden müssen.

1.4 Der aktuelle Wissensstand

Noch nicht vorhandene Kenntnisse über die Anwendung von Clusteralgorithmen auf das AT werden hauptsächlich aus Onlinere Ressourcen bezogen. Primärliteratur zur gewählten Clustersoftware ist unter <http://www.cs.waikato.ac.nz/ml/weka/> zu finden. Unter dieser Adresse wird die Clustersoftware Weka vorgestellt. Auf der Seite wird auch auf das passende Eingabeformate ARFF eingegangen, auf dem dann die Clusteralgorithmen arbeiten.

1.5 Testrechner

Der Testrechner besteht aus einem Laptop mit einem 64 Bit Dual Core Prozessor von AMD mit 2.1 GHz und 8GB RAM.

1.6 Der Aufbau des Alten Testaments

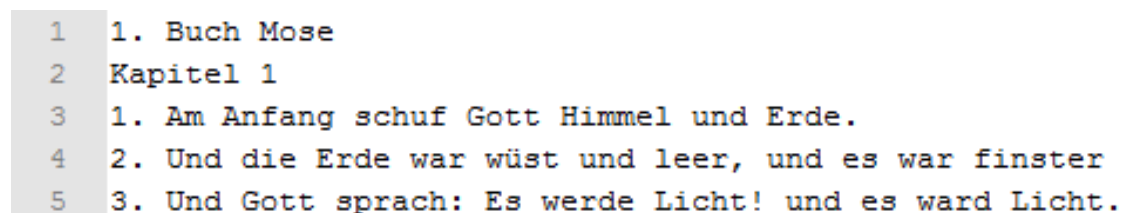
Jeder Vers des AT steht in einer eigenen Zeile, unabhängig, ob der Vers eher kurz ist, wie „Am Anfang schuf Gott Himmel und Erde.“

oder eher lang ist wie: „Da wurden berufen des Königs Schreiber zu der Zeit im dritten Monat, das ist der Monat Sivan, am dreiundzwanzigsten Tage, und wurde

geschrieben, wie Mardochai gebot, an die Juden und an die Fürsten, Landpfleger und Hauptleute in den Landen von Indien bis an das Mohrenland, nämlich hundert und siebenundzwanzig Länder, einem jeglichen Lande nach seiner Schrift, einem jeglichen Volk nach seiner Sprache, und den Juden nach ihrer Schrift und Sprache.“

Die Zeilennummer identifiziert somit einen Vers eindeutig.

Neben den Versen gibt es auch noch Kapitelüberschriften wie z.B. „Kapitel 1“ und Buchnamen wie z.B. „1. Buch Mose“, die auch eine eindeutige Zeilennummer haben. Zur Vereinfachung wird auch bei Buchnamen und Kapiteln von Versen gesprochen. Die Verse eines Buches im AT sind zwar auch durchnummeriert, diese Nummerierung wird aber ignoriert, da sie nur innerhalb eines Kapitels eindeutig ist.



```
1 1. Buch Mose
2 Kapitel 1
3 1. Am Anfang schuf Gott Himmel und Erde.
4 2. Und die Erde war wüst und leer, und es war finster
5 3. Und Gott sprach: Es werde Licht! und es ward Licht.
```

Abbildung 1.1: Aufbau des AT

1.7 ARFF

Damit die Clusteralgorithmen auf das AT angewandt werden können, muss das AT vorher in das ARFF-Format (Attribute-Relation File Format) umgewandelt werden.

ARFF (Attribute-Relation File Format) ist eine Textdatei, die eine Liste von Instanzen beschreibt, die sich eine Menge von Attributen teilen. Eine Instanz wäre in unserem Beispiel ein Vers des AT. Die Attribute, die sich ein Vers mit anderen Versen teilt, sind die Wörter aus denen der Vers besteht. Der Vers „1. Am Anfang schuf Gott Himmel und Erde.“ besteht aus der Attribut-Menge 1, Am, Anfang, schuf, Gott, und, Erde, Himmel

Die Elemente „1“, „Himmel“, „und“, „Erde“ teilt sich der Vers z.B. mit dem Vers „1. Also ward vollendet Himmel und Erde mit ihrem ganzen Heer.“

Man kann sagen, dass diese zwei Instanzen bzw. Verse deswegen eine gewissen Ähnlichkeit haben, da sie sich einige gemeinsame Attribute teilen.

Eine ARFF-Datei besteht aus zwei Teilen. Dem Header- und dem Data-Teil. Im Header-Teil befinden sich die Attribute. Zu beachten ist, dass es sich bei dem Wert eines Attributes um die Häufigkeit handelt, wie oft das Attribut im Vers vorkommt. Jedes Attribute hat einen Datentyp. Im vorliegenden Fall handelt es sich um Zahlen. Im Datenteil befinden sich die Instanzen, d.h. die Verse. Sie bestehen aus den Häufigkeiten, die mit Komma getrennt sind. Eine Instanz im Datenteil hat so viele mit Komma getrennte Zahlen, wie es Attribute im Header gibt. Jede Zahl steht für ein Attribut im Header. Die Position eines Attributes im Header stimmt mit der Position des Attributes in der Instanz überein. D.h. das die dritte Zahl einer Instanz für das Attribut „Anfang“ steht, da dieses auch an dritter Position im Header auftaucht.

Im folgenden sind drei Verse gegeben:

1. Buch Mose

Kapitel 1

1. Am Anfang schuf Gott Himmel und Erde.

Für diese drei Verse werden zehn Attribute definiert:

% 1. Title: AT @RELATION AT

@ATTRIBUTE 1 NUMERIC

@ATTRIBUTE Am NUMERIC

@ATTRIBUTE Anfang NUMERIC

@ATTRIBUTE schuf NUMERIC

@ATTRIBUTE Gott NUMERIC

@ATTRIBUTE Himmel NUMERIC

@ATTRIBUTE und NUMERIC

@ATTRIBUTE Erde NUMERIC

@ATTRIBUTE Fluss NUMERIC

@ATTRIBUTE Feld NUMERIC

Im Daten-Teil wird jeder Vers durch eine Instanz ausgedrückt. Die Instanz hat soviele Zahlen wie es Attribute im Header-Teil gibt. Eine 0 bedeutet, das Attribut kommt im Vers nicht vor. Eine 1 bedeutet, das Attribut kommt im Vers einmal vor.

@DATA

1,0,0,0,0,0,0,0,0

1,0,0,0,0,0,0,0,0

1,1,1,1,1,1,1,1,0,0

1.8 Wie kann das AT in das ARFF-Format überführt werden?

Dazu wird ein Programm in Java namens Text2ARFFConverter entwickelt. Es liest alle Wörter die im AT vorkommen ein, filtert doppelte dabei aus. Dann zählt es, wie häufig die Wörter im AT vorkommen. Es sortiert dann die Wörter nach Häufigkeit. Mit einem Startparameter kann angegeben werden, wie viele Wörter bzw. Attribute in den Header der zu erzeugenden ARFF-Datei geschrieben werden sollen. Wird das Programm mit 100 gestartet, werden bei der Erzeugung der ARFF-Datei die 100 am häufigsten vorkommenden Wörter als Attribute in die ARFF geschrieben. Im Datenteil der ARFF-Datei werden die Instanzen erzeugt. Dazu wird das AT Zeilenweise eingelesen. Wenn sich ein Wort unter den z.B. häufigsten 100 Wörtern befindet, wird gezählt wie oft es im Satz vorkommt. Wurden alle Wörter eines Verses gezählt, kann die Instanz geschrieben werden.

1.9 Analyse Text2ARFFConverter

In Abbildung 1.2 auf Seite 6 ist das Analyseklassendiagramm des Text2ARFF-Converters zu sehen.

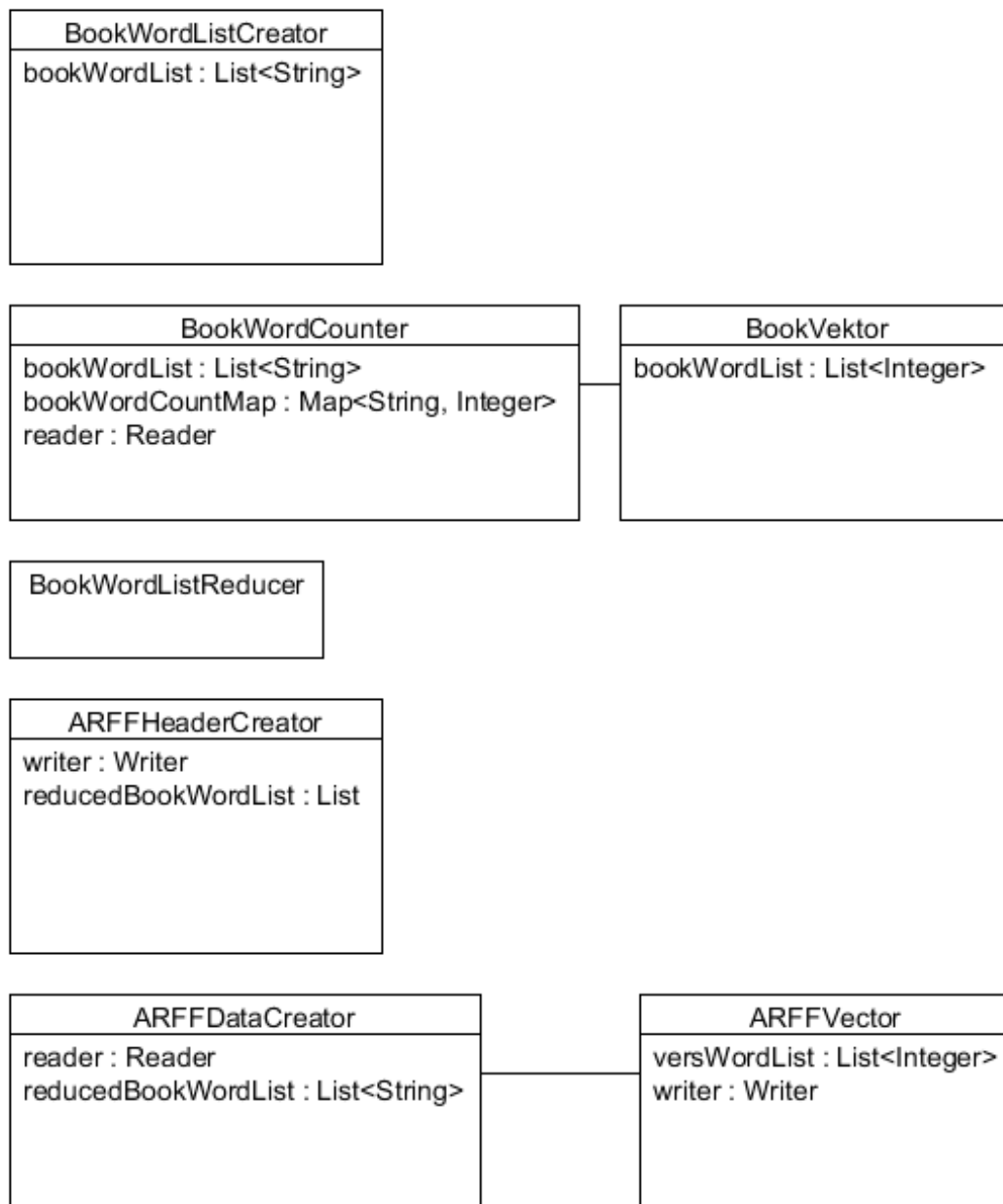


Abbildung 1.2: Analyseklassendiagramm Text2ARFFConverter

1.10 Entwurf Text2ARFFConverter

In Abbildung 1.3 auf Seite 7 ist das Entwurfsklassendiagramm des Text2ARFF-Converters zu sehen.

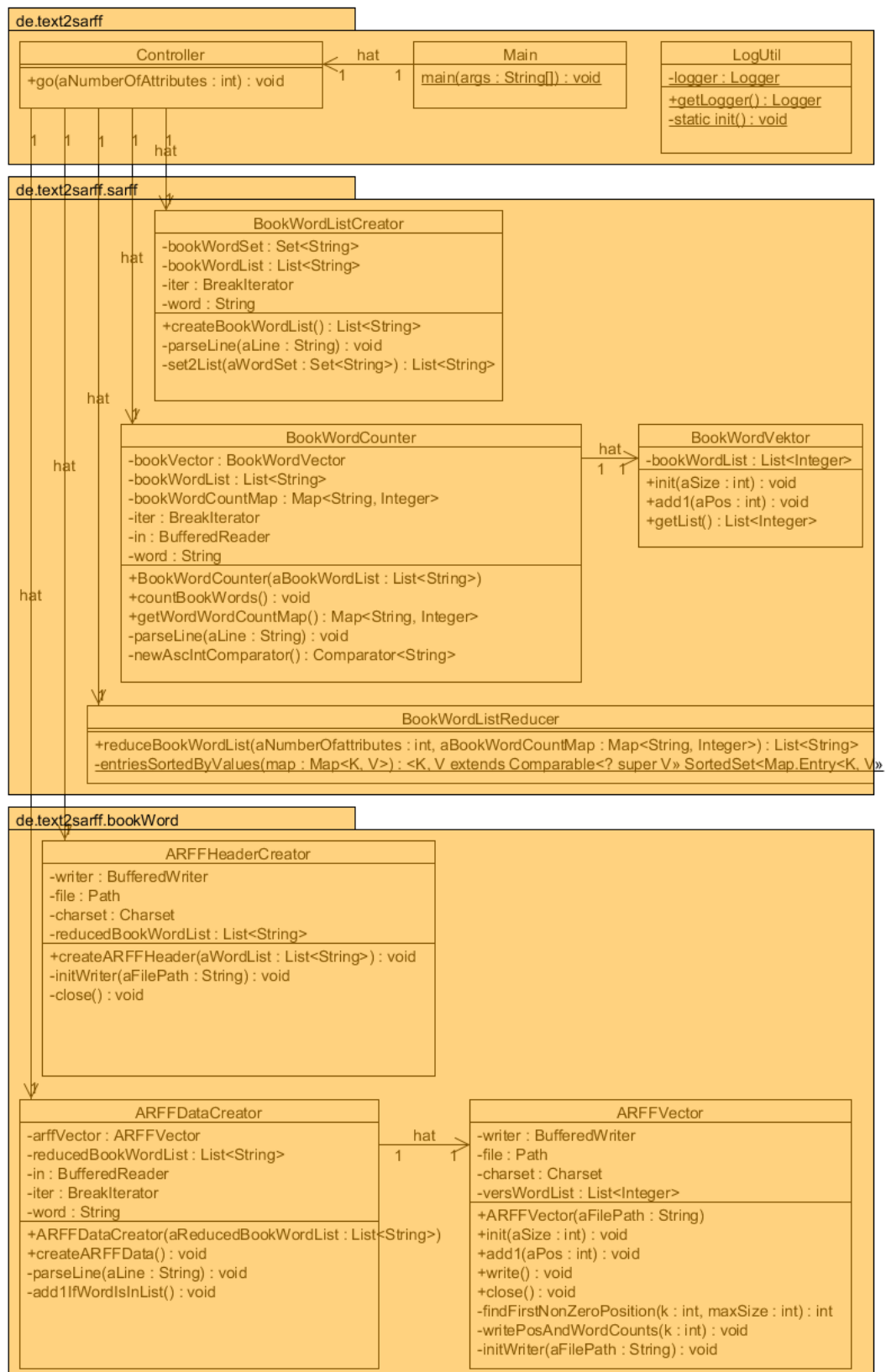


Abbildung 1.3: Entwurfsklassendiagramm Text2ARFFConverter

1.11 Nach welchen Wörter sollte geclustert werden?

Bei Versuchen hat sich gezeigt, dass die Cluster ausgewogener gefüllt sind, wenn die häufigsten Wörter gewählt werden. Werden die seltensten Wörter gewählt, entstehen viele leere Instanzen und es sammeln sich alle Verse in einigen wenigen Clustern und die restlichen Cluster sind fast leer. Wahrscheinlich ist das damit zu begründen, dass beim Clustern ähnliche Verse in ein Cluster kommen. Wenn nun aber durch die seltensten Wörter die Verse alle sehr unterschiedlich sind, wird es für den Algorithmus schwierig, Ähnlichkeiten zu finden. Mit einem Wort, dass nur einmal vorkommt, kann man einen Vers gut von anderen Versen unterscheiden, aber es wird nicht helfen, zu sehen, wo die Ähnlichkeiten zu anderen Versen sind. Wenn dagegen Wörter gewählt werden, die häufig vorkommen, entstehen Cluster, die ausgewogener gefüllt sind. Es kommt mit höherer Wahrscheinlichkeit zwei Verse in dasselbe Cluster, wenn sie an den selben Positionen die gleichen Häufigkeiten haben.

1.12 Hürden beim Einlesen der ARFF-Datei

Die erzeugte ARFF-Datei wird für das AT 900 MB groß. In Weka kommt auf dem Testrechner beim Einlesen der 900MB großen ARFF-Datei die Fehlermeldung „OutOfMemory“. Und Weka hat sich darauf hin geschlossen. In der Konfigurationsdatei RunWeka.ini konnte der Parameter maxheap nur auf höchstens 1550m gesetzt werden. Wird mehr Platz für den heap space angegeben, kommt eine Fehlermeldung von der JVM. Darauf hin wurde von einem 32Bit-Java auf ein 64Bit-Java gewechselt. Damit verschwanden beide Fehlermeldungen und es konnte der maxheap auf 8000m gesetzt werden.

Wird die ARFF-Datei eingelesen, braucht Weka bzw. die JVM dafür 6,8 GB Arbeitsspeicher. Auf dem Entwicklungsrechner stehen aber nur 8GB zur Verfügung. Es wird davon ausgegangen, dass für die Anwendung einiger Clusteralgorithmen der restliche Arbeitsspeicher nicht ausreicht und die Geschwindigkeit durch Swapping ausgebremst wird.

Um die Größe der ARFF-Datei zu reduzieren, gibt es das Format Sparse ARFF, dass auch große Datenmengen kompakt speichern kann.

Sparse ARFF Dateien sind gewöhnlichen AARF Dateien ähnlich, außer das Attribut mit dem Wert 0 nicht repräsentiert werden. Nicht-Null Attribute werden durch die Attributnummer und den Wert angegeben.

Durch Sparse ARFF konnte die Dateigröße von 900MB auf 4 MB reduziert werden. Die Datei wird dadurch auch schneller von Weka eingelesen und Weka braucht weniger RAM. <http://wiki.pentaho.com/display/DATAMINING/>

1.13 SimpleKMeans

Beim Clustern des AT auf dem Testrechner wurden in Abhängigkeit von Attributanzahl und Anzahl der Cluster folgende Ausführungszeiten erzielt.

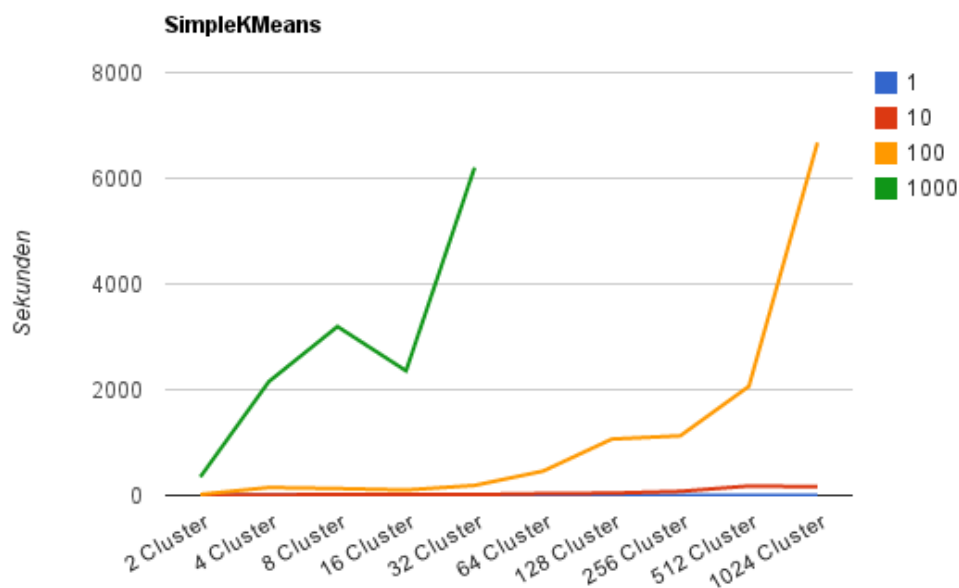


Abbildung 1.4: Ausführungszeiten von SimpleKMeans

In dem Diagramm ist erkennbar, dass die Ausführungszeit steigt, je mehr Attribute und je mehr Cluster es gibt. Auf einem einzelnen Rechner dauert das Clustern des AT für 1000 Attribute und 8 Clustern schon eine knappe Stunde. Für mehr Attribute oder mehr Cluster stößt der Testrechner an seine Grenzen.

1.14 Weitere Cluster-Algorithmen

Neben SimpleKMeans wurde auch XMeans auf das AT angewandt. Bei XMeans grenzt man die Cluster, die er finden könnte mit einer oberen und unteren Schranke ein. XMeans ist ein vergleichsweise schneller Clusteralgorithmus, der das AT selbst mit der maximalen Anzahl von 8950 Attributen in 77 Minuten clustert. Dabei findet er drei Cluster. Als obere Schranke wurden zwei und als obere Schranke wurden 16 Cluster angegeben.

Neben SimpleKMeans und XMeans wurden weitere Clustering-Algorithmen angewandt, die aus verschiedenen Gründen nicht weiter verfolgt wurden. Das Hierarchical Clustering braucht bei nur 5 Attributen mehr als 8GB Arbeitsspeicher um das AT zu clustern. Deswegen konnte dieser Algorithmus auf dem Testrechner nicht ausgeführt werden.

Der EM Algorithmus (expectation maximisation) benötigt auf dem Testrechner bei zehn Attributen 87 Minuten und findet dabei 7 Cluster. Auf ein clustern mit 100 oder gar 1000 Attributen wurde aus Zeitgründen verzichtet.

Der Clusteralgorithmus DBScan bringt beim starten die Fehlermeldung: „Problem evaluating cluster: null“. Auch er wurde nicht weiter verfolgt.

1.15 geclusterte Instanzen wieder in Verse verwandeln

Um aus den geclusterten Instanzen wieder Verse zu machen, die in die jeweils passende Cluster-Datei geschrieben werden, wird ein Programm namens Text2ClusterFile entwickelt. Das Programm gruppiert anhand der Clusternummer und der Zeilennummer die Verse. So wird erkenntlich, welche Verse wie geclustert wurden. In Abbildung 1.5 auf Seite 11 ist das Analyseklassendiagramm von Text2ClusterFile zu sehen.

In Abbildung 1.5 auf Seite 11 ist das Entwurfsklassendiagramm von Text2ClusterFile zu sehen.

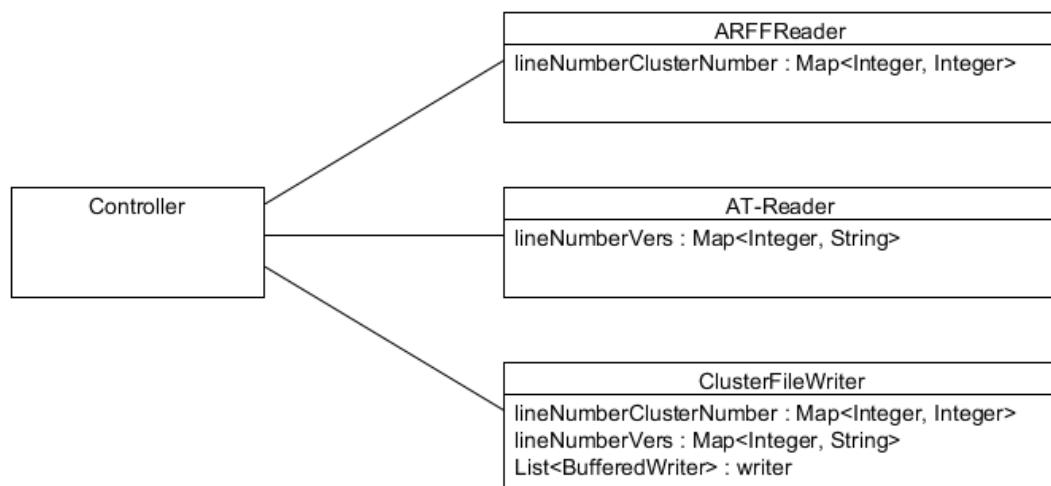


Abbildung 1.5: Analyseklassendiagramm Text2ClusterFile

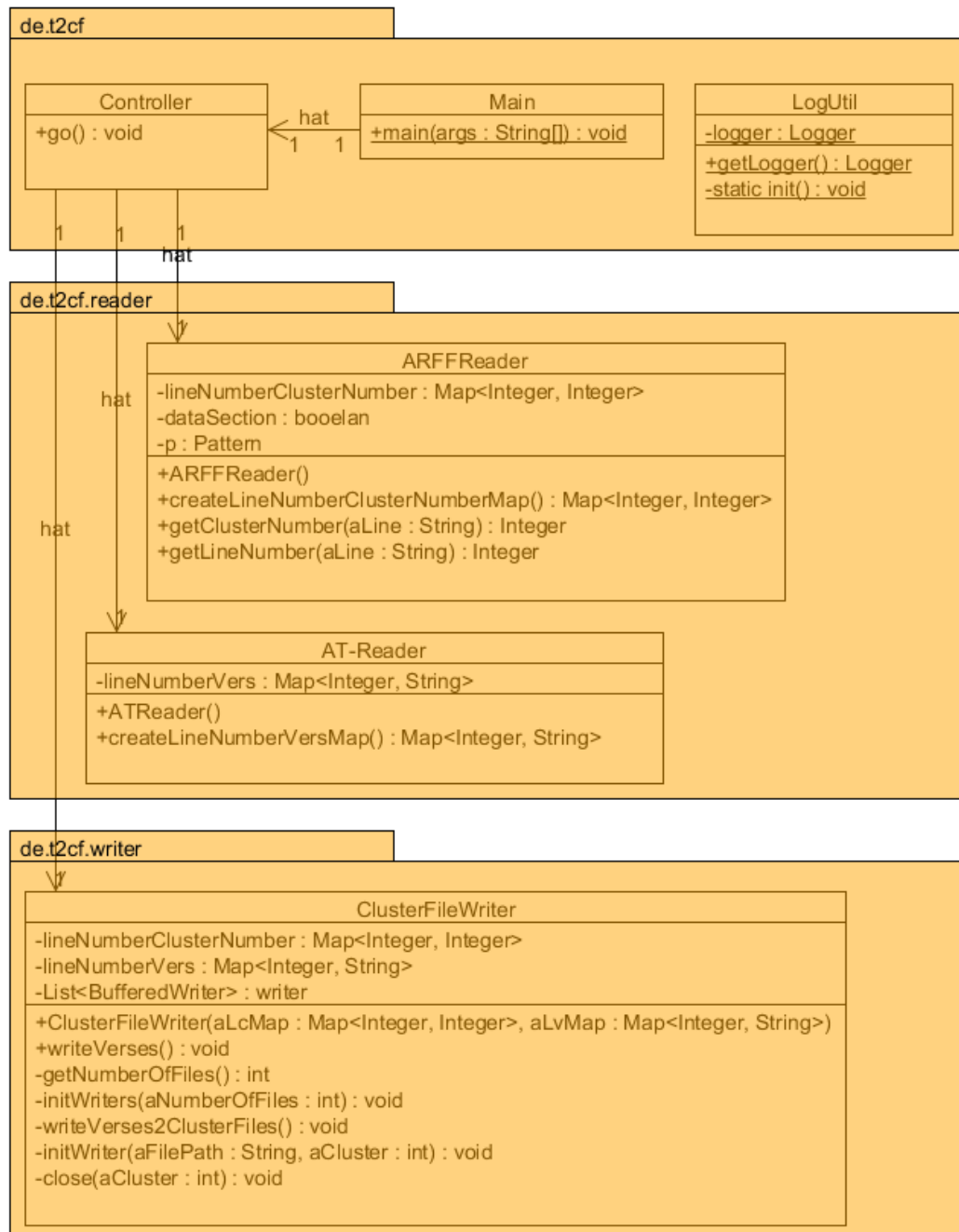


Abbildung 1.6: Entwurfsklassendiagramm Text2ClusterFile

2 Umsetzung

2.1 Zusammenfassung

Beim Clustern des AT nach Versen wurde je nach Algorithmus viel RAM oder viel CPU Zeit benötigt. Der Testrechner ist dabei relativ schnell an seine Grenzen gestoßen. Wird eine größere Menge von Daten geclustert, sollte das nicht nur auf einem einzelnen Rechner erfolgen, sondern verteilt.

2.2 Ausblick

In einem Folgeprojekt könnte mit Werkzeugen wie Apache Mahout verwendet werden, mit dem auch verteilt geclustert werden kann.

A Codebeispiele

Listing A.1: alle Tabellen erstellen

```
1 CREATE TABLE "user"
2 (
3     userid bigint ,
4     name text ,
5     email text ,
6     gender text ,
7     birthday date ,
8     password text ,
9     image text
10 )
11 WITH (
12     OIDS=FALSE
13 );
14 ALTER TABLE "user"
15     OWNER TO postgres;
16
17 CREATE TABLE event
18 (
19     eventid bigint NOT NULL,
20     creatorid bigint ,
21     date date ,
22     eventname text ,
23     occasion text ,
24     location text ,
25     lon double precision ,
26     lat double precision ,
27     description text ,
```

```
28     numbermaleconfirmed int ,
29     numberfemaleconfirmed int
30 )
31 WITH (
32     OIDS=FALSE
33 );
34 ALTER TABLE event
35     OWNER TO postgres;
36
37 CREATE TABLE message
38 (
39     messageid bigint ,
40     eventid bigint ,
41     senderid bigint ,
42     recipientid bigint ,
43     textmessage text ,
44     date date
45 )
46 WITH (
47     OIDS=FALSE
48 );
49 ALTER TABLE message
50     OWNER TO postgres;
51
52 CREATE TABLE participation
53 (
54     participationid bigint ,
55     userid bigint ,
56     eventid bigint
57 )
58 WITH (
59     OIDS=FALSE
60 );
61 ALTER TABLE participation
62     OWNER TO postgres;
```

Listing A.2: Datenimport über COPY

```
1 COPY public.Event (eventid, creatorid, date, eventname,
   occasion, location, lon, lat, description,
   numbermaleconfirmed, numberfemaleconfirmed) From 'C:\
   Event.txt' DELIMITER ';';
2 COPY public.Message (messageid, eventid, senderid,
   recipientid, textmessage, date) From 'C:\Message.txt'
   DELIMITER ';';
3 COPY public.Participation (participationid, userid, eventid
   ) From 'C:\Participation.txt' DELIMITER ';';
4 COPY public.User (userId, name, email, gender, birthday,
   password, image) From 'C:\User.txt' DELIMITER ';';
```

Listing A.3: Primär- und Fremdschlüssel hinzufügen

```
1 ALTER TABLE public.event ADD PRIMARY KEY (eventid);
2 ALTER TABLE public.message ADD PRIMARY KEY (messageid);
3 ALTER TABLE public.participation ADD PRIMARY KEY (
   participationid);
4 ALTER TABLE public.user ADD PRIMARY KEY (userid);
5
6 ALTER TABLE event ADD CONSTRAINT event_creatorid FOREIGN
   KEY (creatorid) REFERENCES public.user (userid) MATCH
   FULL;
7 ALTER TABLE message ADD CONSTRAINT message_eventid FOREIGN
   KEY (eventid) REFERENCES event (eventid) MATCH FULL;
8 ALTER TABLE message ADD CONSTRAINT message_senderid FOREIGN
   KEY (senderid) REFERENCES public.user (userid) MATCH
   FULL;
9 ALTER TABLE message ADD CONSTRAINT message_recipientid
   FOREIGN KEY (recipientid) REFERENCES public.user (userid
   ) MATCH FULL;
10 ALTER TABLE participation ADD CONSTRAINT
   participation_userid FOREIGN KEY (userid) REFERENCES
   public.user (userid) MATCH FULL;
11 ALTER TABLE participation ADD CONSTRAINT
   participation_eventid FOREIGN KEY (eventid) REFERENCES
```

```
event (eventid) MATCH FULL;
```

Listing A.4: Indexe auf Spalten legen

```
1 CREATE INDEX event_creatorid ON public.event(creatorid);
2 CREATE INDEX message_eventid ON public.message(eventid);
3 CREATE INDEX message_senderid ON public.message(senderid);
4 CREATE INDEX message_recipientid ON public.message(
    recipientid);
5 CREATE INDEX participation_userid ON public.participation(
    userid);
6 CREATE INDEX participation_eventid ON public.participation(
    eventid);
7
8 CREATE INDEX event_date ON public.event(date);
9 CREATE INDEX event_eventname ON public.event(eventname);
10 CREATE INDEX event_occasion ON public.event(occasion);
11 CREATE INDEX event_location ON public.event(location);
12 CREATE INDEX event_lon ON public.event(lon);
13 CREATE INDEX event_lat ON public.event(lat);
14 CREATE INDEX event_numbermaleconfirmed ON public.event(
    numbermaleconfirmed);
15 CREATE INDEX event_numberfemaleconfirmed ON public.event(
    numberfemaleconfirmed);
16
17 CREATE INDEX message_textmessage ON public.message(
    textmessage);
18 CREATE INDEX message_date ON public.message(date);
19
20 CREATE INDEX user_name ON public.user(name);
21 CREATE INDEX user_email ON public.user(email);
22 CREATE INDEX user_gender ON public.user(gender);
23 CREATE INDEX user_birthday ON public.user(birthday);
```


B Arbeitsaufteilung

Arbeit	C. Ochmann	I. Körner
Abstract		0
Einleitung		1.1
Aufgabenstellung		1.2
Forschungsgegenstand		1.3
akt. Wissensstand		1.4
Zusammenfassung		2.1
Ausblick		2.2

Tabelle B.1: Aufteilung

C Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe. Mir ist bekannt, dass jede Form des Plagiats mit der Note 5 (Betrugsversuch) bewertet wird.

Ochmann, Christof

Unterschrift:

Körner, Ingo

Unterschrift: