[Studien|Diplom|Bachelor|Master]arbeit

**** Titel der Arbeit ****

Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Lehrstuhl Technologie der Informationssysteme

angefertigt von:	**** eigener Name ****
betreuender Hochschullehrer:	** Name des betreuenden Hochschullehrers **
Betreuer:	** Name des Betreuers **

Kiel, ** Datum der Abgabe **

Aufgabe

Name, Vorname:	** Name, Vorname **
Immatrikulations-Nr:	** Immatrikulations-Nr **
Studiengang:	** Studiengang **
betreuender Hochschullehrer:	** Name des Hochschullehrers **
Betreuer:	** Name des Betreuers **
Institut:	Institut für Informatik
Arbeitsgruppe:	Technologie der Informationssysteme
Beginn am:	** Datum des Beginns **
Einzureichen bis:	** Abgabetermin **

Aufgabenstellung:

** Text der Aufgabenstellung **

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

.....
** eigener Name **

Contents

1. Einführung **1**

2. Zweites Kapitel **3**
2.1. Erster Unterabschnitt 3
2.2. Zweiter Unterabschnitt 3
2.3. Dritter Unterabschnitt 3

3. Concepts **5**
3.1. Question Answer System 5
3.2. Rating System 5
3.3. Tagging 6
3.3.1. Levenshtein Distance 6
3.3.2. Optimized Levenshtein distance algorithm 7
3.4. Recommendation 8
3.4.1. Item-Based Recommendation 9
3.4.2. Collaborative Filtering 9
3.4.3. Singular Value Decomposition 10

A. Erster Anhang **11**

B. Zweiter Anhang **13**

List of Figures

List of Tables

List of Algorithms

1. Recursive Levenshtein Distance Algorithm 6

Abkürzungen

1. Einführung

** Jetzt geht's los! **

2. Zweites Kapitel

2.1. Erster Unterabschnitt

2.2. Zweiter Unterabschnitt

2.3. Dritter Unterabschnitt

3. Concepts

3.1. Question Answer System

3.2. Rating System

3.3. Tagging

The purpose of tagging is to find one or more keywords for a text that describe the content. For the context of economics the *STW Thesaurus for Economics* includes16000 labels which can be used to match words from a text and provide therefore a good tagging base. However the *STW Thesaurus for Economics* only includes the basic form of the words. Therefore a normal test for equal is not sufficient enough. For example a word that is used in it's plural form in a text would return false on a test for equal and would therefore not be a candidate as a tag for the text. This creates the need for a metric that indicates the distance between two words. A practical metric for such a task is the *levenshtein distance*.

check number of labels and add a source

3.3.1. Levenshtein Distance

The *levenshtein distance* calculates the minimum numbers of substitutions, insertions and deletions that are needed to change one word into another. Therefore it returns zero if the words are equal and adds one to the result if it has to perform a substitution an insertion or a deletion of a letter compare algorithm 1.

$$levDist_{a,b}(i,j) = \begin{cases} max(i,j) & \text{if min(i,j)} = 0 \\ min \begin{cases} levDist_{a,b}(i-1,j)+1 \\ levDist_{a,b}(i,j-1)+1 \\ levDist_{a,b}(i-1,j-1)+[a_i \neq b_i] \end{cases} & \text{else} \end{cases}$$

find better solution to move 'if'

This can be written as a recursive algorithm.

Algorithm 1 Recursive Levenshtein Distance Algorithm

```
1:  procedure LevenshteinDistance(s : String, t : String)
2:      lenS ← length(s)
3:      if lenS = 0 then
4:          return lenT
5:      end if
6:      if lenT = 0 then
7:          return lenS
8:      end if
9:      if s[lenS-1] = t[lenT-1] then      ▷ test if last characters of the strings match
10:         cost ← 0
11:     else
12:         cost ← 1
13:     end if
14:     return minimum of
          LevenshteinDistance(s[0..lenS − 1], t) + 1,
          LevenshteinDistance(s, t[0..lenT − 1) + 1,
          LevenshteinDistance(s[0..lenS − 1], t[0..lenT − 1]) + cost)
15: end procedure
```

However the direct implementation of the *levenshtein distance algorithm* has a complexity of O(something). Therefore it is a good utility to better understand the *levenshtein distance* in general, but it is not feasible for a software that should work in production mode.

3.3.2. Optimized Levenshtein distance algorithm

Definition 1 (Non Deterministic Finite Automata)**.** *A finite automata is a tupel of the form* $FA = (Q, \Sigma, q_0, \Delta, F)$*. Q is the set of the states. Σ is the set of the input alphabet. $q_0 \in Q$ is the initial state and $F \subset Q$ is a subset that contains the final states. Δ is a relation of the form $\Delta \subset Q \times \Sigma \times Q$. FA is called finite exactly when Q is finite. Furthermore is $\Sigma*$ the set of words over Σ and ϵ is the empty word.*

Definition 2 (Deterministic Finite Automata)**.** *FA is deterministic if for all $p \in Q$ and all $a \in \Sigma$ exists exactly one state $q \in Q$ with $(p, a, q) \in \Delta$. In this case Δ is written as a function $\delta : Q \times \Sigma \to Q$.*

Definition 3 (Path)**.** *A path for FA is a series $\pi = p_0 a_1 p_1 a_2 \dots a_n p_n$, $(p_i, a_{i+1}, p_{i+1}) \in \Delta$ and $0 \leq i \leq n-1$. The length of π is n and the label for $\beta(\pi)$ is $a_1 a_2 a_3 \dots a_n$.*

Definition 4 (Path shortwriting)**.** *$FA : p \xrightarrow{\omega} q$ with $\omega \in \Sigma*$ states, that a path π for FA from p to q with label $\beta(\pi) = \omega$ exists.*

Definition 5 (FA accepts a word)**.** *FA accepts $\omega \in \Sigma^*$, if and only if $p \in I, q \in F$ exists with FA: $p \xrightarrow{\omega} q$. For FA let $\mathcal{L}(FA) = \{\omega \in \Sigma^* \mid FA \text{ accepts } \omega\}$ be the language that FA accepts.*

Definition 6 (Formal Levenshtein Distance)**.** *The levenshtein distance between two words $V, W \in \Sigma^*$ is the minimal number of edit operations (substitutions, deletions or insertions) that are needed to transform V into W, $d_L(V, W)$ denotes the levenshtein distance between V and W.*

check if I have to add source to the definitions

Definition 7. *$\mathcal{L}_{Lev}(n, W)$, $n \in \mathbb{N}$ and $W \in \Sigma^*$ is the set that denotes all words $V \in \Sigma^*$ such that $d_L(W, V) \leq n$.*

Definition 8 (Degree of FA)**.** *Let $W \in \Sigma^*$ and $n \in \mathbb{N}$. A finite state automaton A is a* Levenshtein automaton *of degree n for W if and only if $\mathcal{L}(A) = \mathcal{L}_{Lev}(n, W)$.*

An optimized version of the *levenshtein distance algorithm* that uses a deterministic finite automata was introduced in *Fast string correction with Levenshtein automata by Schulz and Mihov*. The deterministic finite automata for a word $W \in \Sigma^*$ is a *levenshtein automata* with degree $n \in \mathbb{N}$ with n length of W and has has the following structure:

optimize this

- Σ is the complete alphabet

- Each state in Q denotes for an insert word $\omega \in \Sigma^*$ the number of matching letters together with $d_L(\omega, W)$
- The initial state q_0 is the state with insert count 0 and *levenshtein distance* 0
- The function δ is the known function $\delta : Q \times \Sigma \to Q$.
- The set of final states F contains all states where the correct letter count is equal to the size of the original word W.

definiere delta

The deterministic finite automata with degree $n \in \mathbb{N}$ for a word $W \in \Sigma^*$ can decide in linear time if for a word $V \in \Sigma^*$ $d_L(W, V) \leq n$.
Lemma 6 For any fixed number n, given two words W and V of length w and v respectively, it is decidable in time O(max(w, v)) if the Levenshtein distance between W and V is <= n.
The current system implementation calculates for every word from the text an automata and reads all words from the *STW Thesaurus for Economics*. If a word from the *STW Thesaurus for Economics* ends in a final state it is used as a tag for the text. A possible optimization would be to calculate once all the automata's for the complete *STW Thesaurus for Economics* and to store it with an efficient data structure in a database. Currently the *STW Thesaurus for Economics* is stored in a triple store due to the fact that it can easily be updated with the standard files from the website.

describe word size maximum correction count correlation.

3.4. Recommendation

more general stuff

A recommendation system recommends items to a user.

Definition 9 (User). *$U = \{u_1, u_2, u_3, \ldots, u_n\}$ is a set with u_i users from the recommendation system. With $n, i \in \mathbb{N}$ and $i \leq n$.*

Definition 10 (Item). *$I = \{i_1, i_2, i_3, \ldots, i_n\}$ is a set with i_j items from the recommendation system. With $n, j \in \mathbb{N}$ and $j \leq n$.*

Definition 11 (Rating). *R is an $n \times m$ matrix with $n = |I|$ and $m = |U|$. Furthermore is $r_{n,m}$ a rating for item $n \in I$ and user $m \in U$ with $r_{n,m}$ entry in R and $r_{n,m} \in \{1, 2, 3, 4, 5\}$. If a user $k \in U$ hasn't rated an item $l \in I$ yet the entry $r_{l,k}$ remains empty. $\hat{I}_u$ is a set with all items $i \in I$ where $r_{i,u}$ is empty, $\tilde{I}_u$ is a set with all items i and $r_{i,u}$ is not empty.*

Definition 12 (Prediction). *A prediction is a rating $r_{i,u}$ for item $i \in I$ and user $u \in U$ with $r_{i,v}$ empty entry in R.*

Definition 13 (Recommendation). *Let $n \in \mathbb{N}$ and $u \in U$. A recommendation is a set of n predictions for a user u ordered by the values of the predictions.*

3.4.1. Item-Based Recommendation

The concepts of item-based recommendations was introduced in 2001 by Sarwar et al *Item-Based Collaborative Filtering Recommendation Algorithms*. The idea behind this concept is that recommendations can be calculated based on similar items of the items that a user likes. If $i, \in \hat{I}_u$, $u \in U$, then we can calculate a prediction for i with the similarity between i and all items $j \in \tilde{I}_u$. There are different possible ways to calculate the similarity between two items. However the similarity calculation with the best performance is *adjusted cosine similarity* which is an optimized form of the *cosine similarity*. (see Sarwar et al)

Definition 14 (Cosinus Similarity)**.** *With* $i, j \in \mathbb{N}^n$, $n \in \mathbb{N}$.

$$cos(\overrightarrow{i}, \overrightarrow{j}) = \frac{\overrightarrow{i} \cdot \overrightarrow{j}}{||\overrightarrow{i}||_2 \cdot ||\overrightarrow{i}||_2} \tag{3.1}$$

If $\overrightarrow{i}, \overrightarrow{j}$ are rating vectors, the individual rating behaviour of a user needs to be taken into account to get the correct similarity of $\overrightarrow{i}, \overrightarrow{j}$. This results into the *adjusted cosine similarity*.

Different rating behaviours

Definition 15 (Adjusted Cosine Similarity)**.**

$$sim(i, j) = \frac{\sum_{u \in U} (R_{u,i} - \overline{R_u})(R_{u,j} - \overline{R_u})}{test} \tag{3.2}$$

With the similarities we can calculate the predictions with the following algorithm.

add algorithm

Example: So, item based is a good choice for websites that need fast scalable recommendations, because all the complex calculations can be computed offline. Furthermore the system only needs a subset of all items because a sample of the 25 most similar items are needed to generate good recommendations see Sarwar et al page 8. Moreover it is a tested concept, amazon.com uses item based recommendations for their product recommendations. The described systemuses item based predictions to decrease the scarcity of the rating matrix.

example

how to call the software implementation?

Definition 16 (Scarcity)**.** *Scarcity is . . . It should be minimized in order to . . .*

write this

Therefore all the calculations that are done for the item based algorithm can be computed offline.

3.4.2. Collaborative Filtering

Collaborative filtering is the task of recommending items based on the interest of similar users.

3.4.3. Singular Value Decomposition

Definition 17 (Singular Value Decomposition)**.** ...

write this

The singular value decomposition is a mathematical concept created by .... The interesting part in hinsight of recommendation system is that with the singular value decomposition it is possible to create the best matrix approximation of the original matrix. Therefore the rows of the original matrix that represent the users can be approximated by a single two dimensional matrix. Thus it is easily possible to calculate the similarity between users. Forexample with cosinus similarity. The calculation of the similar users can be created offline and theirefore fits with the item based recommendation technique. So it is possible to create a pipeline architecture that starts with the item based technique to decrease the scarcity of the user item matrix and if this has finished can calculate the similarity between users. All this information can be stored in a database and can be used to calculate user recommendations if they are needed.

quote proof

A. Erster Anhang

B. Zweiter Anhang