Bachelorarbeit

zum Erlangen des akademischen Grades Bachelor of Science
des Fachbereichs Elektrotechnik/Informatik
der Universität Kassel

Konzept und Umsetzung eines Tag-Recommenders für Video-Ressourcen am Beispiel UniVideo

---

Vorgelegt von:	Sebastian Böttger
Matrikelnummer:	24204510
geboren am:	15. Januar 1983
Erstgutachter:	Prof. Dr. Gerd Stumme
Zweitgutachter:	Prof. Dr. Claudia Fohry
Betreuer:	Dr. Robert Jäschke

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die von mir am heutigen Tage dem Studienservice des Fachbereichs Elektrotechnik/Informatik eingereichte Bachelorarbeit zum Thema

Konzept und Umsetzung eines Tag-Recommenders für
Video-Ressourcen am Beispiel UniVideo

vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Kassel, den 17. April 2012

Unterschrift

Kurzfassung

Kollaborative Verschlagwortungssysteme bieten Nutzern die Möglichkeit zur freien Verschlagwortung von Ressourcen im World Wide Web. Sie ermöglichen dem Nutzer beliebige Ressourcen mit frei wählbaren Schlagwörtern – so genannten Tags – zu versehen (Social Tagging). Im weiteren Sinne ist Social Tagging nichts anderes als das Indexieren von Ressourcen durch die Nutzenden selbst. Dabei sind die Tag- Zuordnungen für den einzelnen Nutzer und für die gesamte Community in vielerlei Hinsicht hilfreich. So können durch Tags persönliche Ideen oder Wertungen für eine Ressource ausgedrückt werden. Außerdem können Tags als Kommunikationsmittel von den Nutzern oder Nutzergruppen untereinander verwendet werden. Tags helfen zudem bei der Navigation, beim Suchen und beim zufälligen Entdecken von neuen Ressourcen. Das Verschlagworten der Ressourcen ist für unbedarfte Anwender eine kognitiv anspruchsvolle Aufgabe. Als Unterstützung können Tag-Recommender eingesetzt werden, die Nutzern passende Tags vorschlagen sollen.

UniVideo ist das Videoportal der Universität Kassel, das jedem Mitglied der Hochschule ermöglicht Videos bereitzustellen und weltweit über das WWW abrufbar zu machen. Die bereitgestellten Videos müssen von ihren Eigentümern beim Hochladen verschlagwortet werden. Die dadurch entstehende Struktur dient wiederum als Grundlage für die Navigation in UniVideo.

In dieser Arbeit werden vier verschiedene Ansätze für Tag-Recommender theoretisch diskutiert und deren praktische Umsetzung für UniVideo untersucht und bewertet. Dabei werden zunächst die Grundlagen des Social Taggings erläutert und der Aufbau von UniVideo erklärt, bevor die Umsetzung der vier einzelnen Tag-Recommender beschrieben wird. Anschließend wird gezeigt wie aus den einzelnen Tag-Recommendern durch Verschmelzung ein hybrider Tag-Recommender umgesetzt werden kann.

Abkürzungsverzeichnis

AJAX Asynchronous JavaScript and XML

CSS Cascading Style Sheets

FLV Flash Video

JSON JavaScript Object Notation

OCR Optical Character Recognition

PHP PHP: Hypertext Preprocessor

REST Representational State Transfer

URL Uniform Resource Locator

WWW World Wide Web

XML Extensible Markup Language

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

Inhaltsverzeichnis

1 Einleitung

Das World Wide Web (WWW), als ein wesentlicher Teil des Internets, ermöglicht es heute jedem Informationen bereitzustellen, abzurufen und zu nutzen. In der ersten Zeit nach Entstehung des WWW war es einigen wenigen Experten vorbehalten Informationen zur Verfügung zu stellen, da besondere Fähigkeiten und Kenntnisse Voraussetzungen dafür waren. In den frühen 2000er Jahren markierte der Begriff Web 2.0 eine Zäsur in der Nutzung des WWW. Es wurde nun für mit dem Web unerfahrene Nutzer auf einfache Weise möglich, selbst Informationen für andere im Web zur Verfügung zu stellen. Neue Techniken ermöglichten ihnen das Bereitstellen von Texten, Bildern und Multimedia-Inhalten – meistens mit „sozialer Software“, also im Austausch mit anderen Nutzern. Dieser *User Generated Content* führte zu einem immensen Anstieg der im Web zur Verfügung gestellten Inhalte. Um diese Mengen an Informationen nutzbar zu machen, entwickelten sich neue Techniken zur Indexierung der Inhalte, die nun auch von „Laien“ durchgeführt werden musste.

Eine Möglichkeit einer Indexierung wird durch das Tagging in so genannten Folksonomien erreicht. Es ermöglicht Benutzern, beliebige Ressourcen mit frei wählbaren Schlagwörtern – so genannten Tags – zu versehen. Diese Zuordnungen sind in vielerlei Hinsicht hilfreich. So können durch Tags persönliche Ideen oder Wertungen für eine Ressource ausgedrückt werden. Außerdem können Tags als Kommunikationsmittel mehrerer Nutzer verwendet werden. Sie helfen bei der Navigation, beim Suchen und beim zufälligen Entdecken von neuen Ressourcen. Mit den Vorteilen, die durch das Indexieren durch die Nutzenden entstehen, geht auch eine Gefahr einher: Ist eine Ressource nicht richtig oder nicht umfangreich genug verschlagwortet worden, wird diese von anderen Nutzern möglicherweise nicht gefunden. Der Sinn der Folksonomy, Informationen mit anderen zu teilen, kann dadurch unbewusst unterlaufen werden. Die gesamte Folksonomy wird durch Inkonsistenzen im Tag-Vokabular selbst inkonsistent und weniger leicht nutzbar. Eine Lösung für dieses Problem können Schlagwortempfehlungssysteme – so genannte Tag-Recommender – sein, die einerseits den Nutzern bei der kognitiv anspruchsvollen Aufgabe der Indexierung behilflich sind, aber auch eine Vereinheitlichung des Tag-Vokabulars ermöglichen können und dadurch die angesprochenen Probleme zu reduzieren helfen.

UniVideo ist das Videoportal der Universität Kassel, das jedem Mitglied der Hochschule ermöglicht im Rahmen von Forschung und Lehre Videos bereitzustellen und weltweit über das WWW abrufbar zu machen. Um den Aufwand zum Betreiben von UniVideo

gering zu halten, ist es als eine hierarchielose Webanwendung realisiert worden. Es gibt für Moderatoren keinerlei Möglichkeiten zur Kategorisierung der Videos. Lediglich der Nutzer selbst muss beim Hochladen jedes Video mit Schlagwörtern versehen. UniVideo ist also eine Folksonomy. Die Ordnung, die durch das Verschlagworten entsteht, dient als Grundlage für die Navigation in UniVideo.

In dieser Arbeit wird untersucht welche Verfahren eines Tag-Recommenders für UniVideo geeignet sind und wie diese praxistauglich umgesetzt wurden.

1.1 Motivation

Seit dem Jahr 2008 arbeite ich im IT-Servicezentrum (ITS[1]) der Universität Kassel als studentische Hilfskraft. Während dieser Tätigkeit wurde mir u. a. die Aufgabe zugeteilt, eine Web-Benutzerschnittstelle für den im ITS genutzten Helix-Server[2] zu programmieren. Diese ermöglicht ein komfortables Bereitstellen von Video-Ressourcen und eine Darstellung der Videos auf einer eigenen Website. Aus einer Sammlung von PHP-Skripten wurde eine umfangreiche serverseitige Lösung entwickelt, die – neben der Bereitstellung von RealMedia[3]-Videos – Video-Clips verschiedenster Formate entgegennimmt, umwandelt und ermöglicht, diese im FLV-Format über einen Flash-Player abzuspielen. Dies kann als Geburtsstunde von UniVideo betrachtet werden. Die Anzahl der Videos stieg stetig an und erforderte eine gewisse Strukturierung. Es lag also nahe, eine Datenbank zu verwenden, die es zum einen ermöglichte Metadaten wie Titel und Beschreibung zu speichern und zum anderen die Ansammlung der verschiedenen Videos der unterschiedlichen Nutzer zu strukturieren. Jedes Video wurde daraufhin in der Datenbank mit Hilfe von Relationen seinem Besitzer zugeordnet. Auf diese Weise ergab sich die Möglichkeit zur Darstellung mehrerer Sichten, die über eindeutige URLs abrufbar sind:

- die Startseite, die alle Videos absteigend zeitlich-sortiert auflistet,
- die Video-Sicht, die das Video und seine Metadaten präsentiert und
- die Benutzer-Sicht, die alle Videos eines Benutzers auflistet.

Die nächste Erweiterung beinhaltete eine Tagging-Funktion. Mit dieser können die Benutzer ihre jeweiligen Videos individuell klassifizieren. Durch das Tagging können sie ihre Videos beim Hochladen dem System bekannt machen, indem sie jedes Video mit einem oder mehreren Tags (Schlagwörtern) annotieren. Hierdurch kam eine weitere Sicht hinzu, welche alle Videos eines bestimmten Tags auflistet. In der Video-Sicht sind seitdem die Tags als Links aufgelistet, mit denen das Video annotiert ist und auf der

[1] `http://www.uni-kassel.de/its`
[2] Software zum Streamen von Audio- und Videodaten
[3] Sammelbezeichnung für Dateiformate des Software-Herstellers RealNetworks

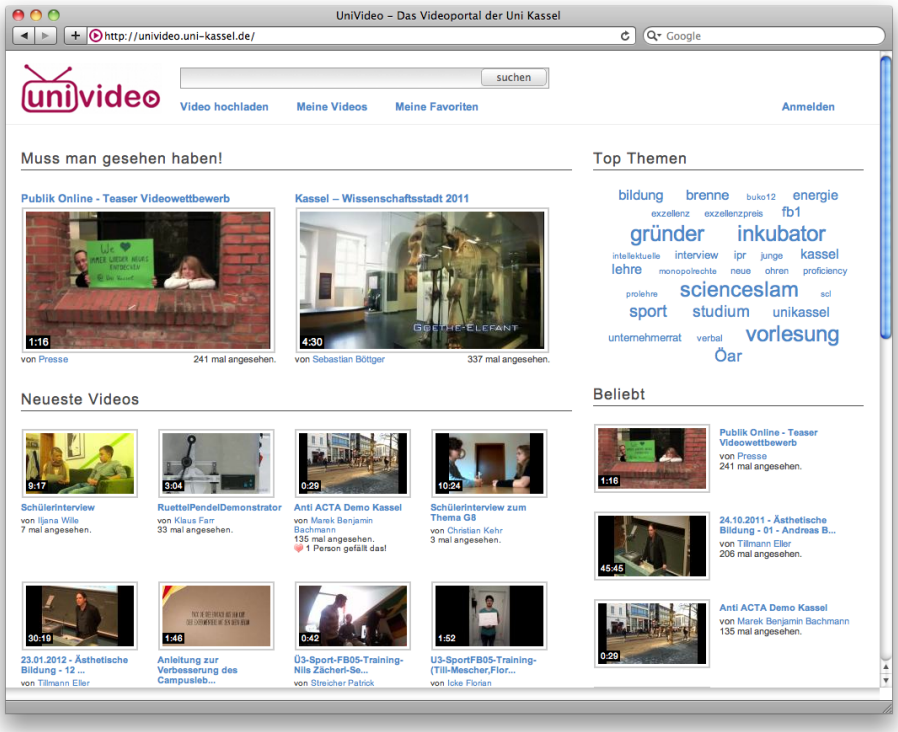


Abb. 1.1: Screenshot UniVideo-Startseite

Startseite wurde an prominenter Stelle eine *Tag Cloud* platziert (siehe Abb. 1.1). Eine *Tag Cloud* ist eine Liste mit den meist verwendeten Tags, wobei die Tags mit der häufigsten Verwendung typografisch hervorgehoben sind. Die Tag Cloud ist also die Darstellung einer Teilmenge der *Folksonomy*. Die einzelnen Tags in der Tag Cloud sind wiederum verlinkt zu ihrer jeweiligen Tag-Sicht. Über die Tags werden für die Video-Sicht ähnliche Videos vorgeschlagen, die u. a. über die Tags ermittelt werden. Es lässt sich feststellen, dass die Tags ein zentrales – oder sogar das zentrale – Navigationselement in UniVideo sind.

Die Motivation einen *Tag-Recommender* – also ein Schlagwortempfehlungssystem – zu entwickeln, liegt in der Bedeutung des Taggings für UniVideo und ist der Tatsache geschuldet, dass die Nutzer nicht immer optimale Tags eingeben, dass ihnen in dem Moment, in dem sie die Videos hochladen, keine passenden Schlagwörter einfallen oder ihnen bestimmte semantische Verbindungen gar nicht bewusst sind. Ich erhoffe mir, dass dem Benutzer – durch die vorgeschlagenen Tags – andere passende Begriffe in den Sinn kommen und somit das Vokabular der Folksonomy erweitert wird. Der Recommender soll dem Benutzer also keinesfalls die Arbeit abnehmen, sondern vielmehr beim Taggen

unterstützend Hilfestellung geben. Auch erhoffe ich mir, dass der Sinn des Taggens dem unbedarften Nutzer ersichtlicher wird. Der Tag-Recommender soll ihn aktivieren und motivieren.

1.2 Gliederung

In Kapitel 2 werden zunächst die Grundlagen des Taggings erläutert. Es werden formal die Begriffe und mathematischen Definitionen des Taggings eingeführt. Auch wird Tagging aus Sicht der Nutzer erläutert, welche Bedeutung es für diese hat und welche Probleme aus dieser Nutzer-Perspektive entstehen. Darauffolgend werden Tag-Recommender formal eingeführt und drei unterschiedliche Quellen zur Generierung von Tag-Vorschlägen vorgestellt und die Vor- und Nachteile, die diese mit sich bringen. In Kapitel 3 wird zunächst knapp zusammengefasst die Funktionsweise und der Aufbau von UniVideo beschrieben. Darauffolgend wird das Framework vorgestellt, auf dessen Basis die Algorithmen für Tag-Recommender umgesetzt und evaluiert werden. Der anschließende Teil befasst sich mit den Benutzerschnittstellen und der Client-Server-Kommunikation. In Kapitel 4 werden die Algorithmen vorgestellt, die für UniVideo entwickelt worden sind und deren praxistaugliche Umsetzung für UniVideo. Die Ergebnisse der Offline- sowie Online-Evaluation werden in Kapitel 5 diskutiert. Das Fazit in Kapitel 6 fasst die Ergebnisse zusammen und gibt einen Ausblick für weitere Optimierungsmöglichkeiten.

2 Grundlagen

Tagging ist zu einem wichtigen Bestandteil moderner Anwendungen im WWW geworden. Es ist eine freie Form der gemeinschaftlichen Indexierung von Ressourcen wie Blogeinträgen, Fotos, Videos oder Lesezeichen. Dabei gibt es keine Regeln – dem Nutzer steht es frei, wieviele und welche Wörter er zur Sacherschließung des Inhalts heranzieht. Da in den meisten Fällen die Ressourcen von mehreren Nutzern verschlagwortet werden können und die gesamte Community Vorteile daraus erzielt, wird in diesem Zusammenhang auch von *Social Tagging* oder *Collaborative Tagging* gesprochen.

In diesem Kapitel werden die Begriffe, die dem *Social Tagging* zugrunde liegen eingeführt und erläutert. Des Weiteren werden die Bedeutung und die Motive des Taggens aus Sicht des Nutzers diskutiert. Anschließend werden einige Grundkonzepte für Tag-Recommender-Systeme vorgestellt und gezeigt, wie sich diese evaluieren lassen.

2.1 Social Tagging

Tags können im weitesten Sinne als das elektronische Äquivalent der Post-Its betrachtet werden [MP08]. Um in einem Stapel Bücher bestimmte Informationen schnell wiederfinden zu können, versieht man diese typischerweise mit kleinen, leicht lösbaren Klebezetteln, welche mit Schlagwörtern beschriftet sind. So wird die große Menge an Informationen strukturiert. Ein ähnliches Motiv dürfte die Erfinder des Social-Bookmarking-Systems Delicious[1] (ehemals del.icio.us) angetrieben haben. Mit dieser Webanwendung wurde die Idee des kollaborativen Verschlagwortens im Web das erste Mal angewandt. Benutzer haben die Möglichkeit, Lesezeichen zu speichern und mit Tags zu versehen, um so besuchte Websites schnell wiederzufinden. Die stetig wachsende Menge an Informationen im World Wide Web wird so für den Benutzer ein Stück beherrschbarer.

Tags finden häufig Anwendung in sozialer Software, mit deren Hilfe die Benutzer auf irgendeine Art, aber meist im WWW, interagieren oder in Beziehung stehen können. Ein Tag ist dabei nicht weiter als ein Term (ein Wort), das von einem Benutzer einer Ressource zugeordnet wird. Die Zuordnung der Tags kann als Tripel (u, r, t) formalisiert werden, bestehend aus einem Benutzer u, einer Ressource r und einem Term t.

[1] `http://www.delicious.com`

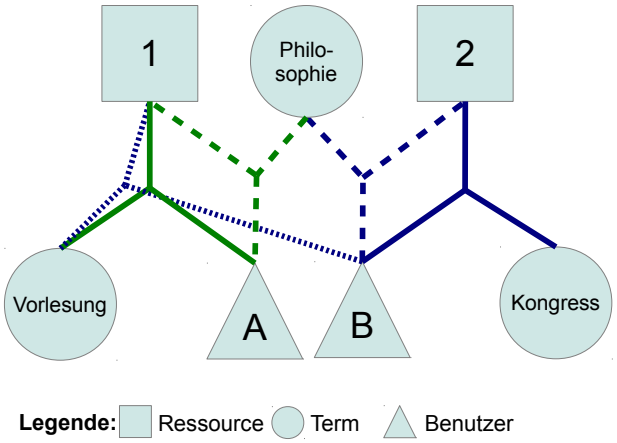


Abb. 2.1: Veranschaulichung als tripartiter Hypergraph

Definition 2.1 *Eine Tag-Zuordnung ist ein Tripel* (u, r, t) *mit* $u \in \{u_1, \ldots, u_j\}$, $r \in \{r_1, \ldots, r_k\}$ *und* $t \in \{t_1, \ldots, t_m\}$

Der entscheidende Vorteil des Taggings gegenüber einer klassischen Kategorisierung ist die Möglichkeit, mehrere Terme einer Ressource zuzuordnen. Der Benutzer muss sich also nicht mehr entscheiden, ob die Ressource der *Kategorie A* oder *B* zugeordnet wird, er kann, sofern es Sinn ergibt, der *Ressource X* das Tag *A* und das Tag *B* zuordnen und ein anderer Benutzer kann wiederum die gleiche *Ressource X* mit einem weiteren Tag *C* versehen, das er möglicherweise zuvor bereits der *Ressource Y* zugeordnet hat. Während sich das Einteilen von Daten in bestimmten Ordnern, wie wir das vom heimischen Rechner kennen, in einer monohierarchischen Baumstruktur abbilden lässt, so führt das Tagging zu einem ungerichteten tripartiten Hypergraphen – also einen Graphen dessen (Hyper-)Kanten immer drei Knoten miteinander verbinden.

Abbildung 2.1 zeigt ein Beispiel eines solchen Hypergraphen. Drei Terme wurden von zwei verschiedenen Benutzern unterschiedlich genutzt. Der *Benutzer A* hat der *Ressource 1* zwei Tags zugeordnet, *Vorlesung* und *Philosophie*. *Benutzer B* hat dieser Ressource nur den Tag *Vorlesung* gegeben, dafür hat er der *Ressource 2* neben *Philosophie* noch den Tag *Kongress* zugeordnet. Um die Gesamtzahl der Tag-Zuordnungen zu erhalten, müssen nur die Hyperkanten gezählt werden, die jeweils einen Benutzer mit einer Ressource und einem Tag verbinden. In diesem Beispiel gibt es also fünf Tag-Zuordnungen.

Diese dreidimensionale Datenstruktur aus Benutzern, Ressourcen und Tags, die einem kollaborativen Tagging-System zugrunde liegen, wird Folksonomy genannt [Wal07]. Das Wort Folksonomy ist ein so genanntes *Kofferwort* bestehend aus *folk* und *taxonomy* und wurde von Thomas Vander Wal geprägt. Er selbst definiert eine Folksonomy so:

> Folksonomy is the result of personal free tagging of information and objects (anything with a URL) for one's own retrieval. The tagging is done in a social environment (usually shared and open to others). Folksonomy is created from the act of tagging by the person consuming the information.
>
> The value in this external tagging is derived from people using their own vocabulary and adding explicit meaning, which may come from inferred understanding of the information/object. People are not so much categorizing, as providing a means to connect items (placing hooks) to provide their meaning in their own understanding. [Wal07]

Mathematisch betrachtet kann als Folksonomy das bezeichnet werden, was der tripartite Hypergraph aus Abbildung 2.1 darstellt.

Definition 2.2 *Eine Folksonomy ist ein Tupel* $\mathbb{F} := (U, R, T, Y)$ *mit*

- *den endlichen Mengen* $U := \{u_1, \ldots, u_j\}$, $R := \{r_1, \ldots, r_k\}$ *und* $T := \{t_1, \ldots, t_m\}$
- *und der ternären Relation* $Y \subseteq U \times R \times T$ *zwischen den Elementen dieser Mengen*

Damit lässt sich der ungerichtete tripartite Hypergraph als Repräsentation der Folksonomy mathematisch wie folgt beschreiben:

Definition 2.3 *Der Graph einer Folksonomy ist ein Tupel* $G := (V, E)$ *mit*

- *einer Menge von Knoten* $V := U \dot\cup R \dot\cup T$
- *und den Hyperkanten* $E := \{\{u, r, t\} \mid (u, r, t) \in Y\}$ *zwischen den Knoten.*

Eine Folksonomy ist demnach die Menge aller Benutzer, Ressourcen und Terme (Entitäten), sowie die Menge aller ternären Assoziationen zwischen den Elementen der Entitätsmengen. Da die Benutzer über die gemeinsam verschlagworteten Ressourcen in Beziehung zueinander stehen, kann man von einem sozialen Netzwerk sprechen.

Eine Folksonomy lässt sich leicht in einer relationalen Datenbank abbilden. Benutzer werden typischerweise durch ihre Nutzer-ID beschrieben, und die Terme als beliebige Zeichenketten. Was eine Ressource ist, hängt vom jeweiligen System ab. In Social-Bookmarking-Systemen wie BibSonomy[2] ist die Ressource eine URL [Jäs11]. In UniVideo ist es ein Video, repräsentiert durch seine eindeutige ID. Durch die inhärente Struktur einer Folksonomy lassen sich Tag-Zuordnungen in einer Datenbank durch die Fremdschlüssel der Entitäten abbilden.

Vander Wal unterscheidet außerdem breite und enge Folksonomien [Wal05]. In einer breiten Folksonomy (Broad Folksonomy) können mehrere Nutzer Terme aus ihrem

[2] `http://www.bibsonomy.org/`

Vokabular an dieselbe Ressource vergeben. Dabei muss die erste Person die Ressource erstellt haben. Die anderen teilen sich dann diese Ressource mit dem Ersteller. Beispiele für breite Folksonomien sind typischerweise Bookmarking-Systeme wie BibSonomy. Bei engen Folksonomien (Narrow Folksonomy) können nur kleine Gruppen von Personen, in den meisten Fällen sogar nur eine Person, derjenige der sie zuerst angelegt hat, Terme an eine Ressource heften. Enge Folksonomien werden meist auf Ressourcen mit binären Inhalten angewandt, deren Content mit einer Textsuche nicht gefunden werden kann. Die Tags werden als wichtige Metadaten benutzt, die das Auffinden dieser Inhalte erleichtern sollen. UniVideo ist nach dieser Definition eine enge Folksonomy, lediglich der Nutzer, der das Video hochgeladen hat, kann dieses auch taggen.

2.2 Tagging aus Nutzersicht

Für die meisten Nutzer ergibt sich der Vorteil des Taggens nicht unmittelbar. Erst mit zunehmender Nutzung bildet sich ein Verständnis dafür und der individuelle Mehrwert wird dem Anwender ersichtlich [MP07]. Hilfreich sind Tags beim Suchen und Finden von Ressourcen. Die Tags werden bei der Berechnung einer Ergebnismenge zu einer bestimmten Suchanfrage mitbenutzt. Außerdem wurde festgestellt, dass kooperatives Verschlagworten zu einer „Veränderung des Suchvorganges von der direkten Suche“, über die Eingabe einer Suchanfrage, „zu einem navigationsbasierten Finden von Informationen“ führt [BBHS08]. Deutlich wird dies an einer Tag-Cloud, welche durch die typographischen Hervorhebungen des (Teil-)Informationsraumes, die am häufigsten verwendeten Terme hervorhebt.

Als Motive zur Nutzung von Tagging-Diensten lassen sich sowohl individuelle (suchen, kategorisieren, finden) und gruppenbezogene (teilen, Aufmerksamkeit auf eigene Ressourcen lenken) Wissensorganisation, als auch individueller (Ideen, Erinnerungen und Kontexte dokumentieren) und sozialer (Beschreibungen und Wertungen abgeben, Gruppenzugehörigkeit ausdrücken) Interpretationshintergrund ausmachen.

Diese Motive können in eine soziale und funktionale Sicht eingeteilt werden [PG08]. Funktional gesehen lassen sich die Kategorisierungseigenschaften des Taggens und die dadurch resultierende Möglichkeit des leichten Teilens von Ressourcen als Organisationselemente zusammenfassen. Dahingegen lassen sich die Möglichkeiten zur Annotierung persönlicher Ideen, Merkhilfen und Erinnerungen sowie die Verwendung von Tags, um Wertungen oder aufgabenbezogene Einschätzungen auszudrücken, unter Kommunikation gruppieren. Aus sozialer Sicht sprechen Panke und Gaiser von einer „Vermischung von individuellem Mehrwert und dem impliziten Nutzen der Schlagwortvergabe für eine Community“. In Anlehnung zu einer Studie von Ames und Naaman [AN07] haben beide Autorinnen ein passendes Schaubild (siehe Abb. 2.2) erstellt, welches die unterschiedlichen Motive beider Aspekte zusammenführt.

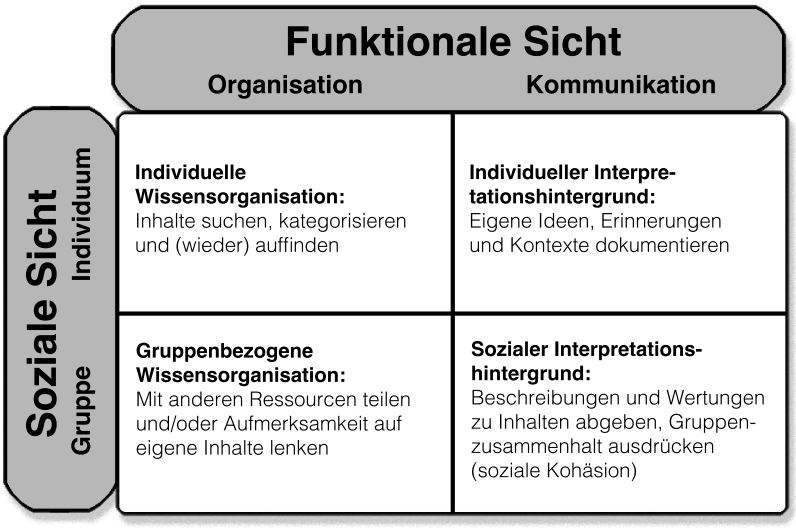


Abb. 2.2: Motive für die Nutzung von Tagging-Diensten. Nachbildung aus [PG08]

Trotz alledem ist Tagging kein Selbstzweck. Das Verschlagworten von Ressourcen ist vielmehr notwendiges Beiwerk zur ursprünglichen Tätigkeit und dem eigentlichen Sinn des jeweiligen Systems. Der Aufwand des Taggens sollte für den Nutzer möglichst gering sein, um ihn nicht vom Kontext seiner eigentlichen Tätigkeit abzulenken [MP08]. Dem Vorgang des Taggens sollte der Nutzer also nicht mehr Aufmerksamkeit schenken, als den Ressourcen die getagged werden. Dem entsprechend sollten auch die Benutzerschnittstellen gestaltet sein.

2.3 Tag-Recommender in Folksonomien

Im Allgemeinen empfehlen Recommender-Systeme dem Benutzer für ihn möglicherweise interessante Informationen oder interessante Produkte. Der Online-Handel Amazon[3] beispielsweise empfiehlt auf der Seite eines Artikels andere Artikel mit dem Hinweis „Kunden, die diesen Artikel gekauft haben, kauften auch“ und zeigt Artikel die „oft zusammen gekauft“ werden. In Tag-Recommender-Systemen sind die Empfehlungen eine Auswahl von Tags, abhängig vom Benutzer und der Ressource.

Definition 2.4 *Tag-Empfehlungen sind eine Auswahl an Tags* $\widetilde{T}(u, r) \subseteq T$*, abhängig vom Benutzer* $u \in U$ *und der Ressource* $r \in R$*.*

[3] `http://www.amazon.com`

In den meisten Fällen werden die vorgeschlagenen Tags $\widetilde{T}(u, r)$ durch eine Rangliste bestimmt, die nach einem bestimmten Qualitäts- oder Relevanz-Kriterium erstellt wird [JMH+07].

2.3.1 Populäre Tags

Die Idee eines Tag-Recommenders ist es, Terme vorzuschlagen, die zu einer bestimmten Ressource als Tags passen könnten. Dazu muss im Umkehrschluss die Nützlichkeit eines bestimmten Terms ermittelt werden können. Ein einfacher Ansatz ist es, davon auszugehen, dass die Terme die häufig als Tags genutzt werden potentiell nützlich sein könnten. Dieser Ansatz ist auch in Bezug auf die Rechenzeit sehr günstig und deswegen gut geeignet für eine Online-Berechnung [Jäs11]. Hierfür wird die Menge aller Tag-Zuordnungen eines Benutzers $Y_u := Y \cap (\{u\} \times T \times R)$ und analog dazu die Menge aller Tag-Zuordnungen einer Ressource Y_r sowie die Menge aller Zuordnungen eines Tags Y_t definiert. Auf ähnliche Weise wird die Menge $Y_{u,t} := Y \cap (\{u\} \times \{t\} \times R)$ beziehungsweise $Y_{t,r} := Y \cap (U \times \{t\} \times \{r\})$ definiert.

Jäschke zeigt in [Jäs11] drei Möglichkeiten die populärsten Tags zu ermitteln:

1. Die Menge der *populärsten Tags der Folksonomy* auszuwählen, ist der einfachste Weg für einen Recommender. Egal für welchen Benutzer oder welche Ressource die Empfehlungen generiert werden sollen, die Ergebnismenge ist die gleiche:

$$\widetilde{T}(u, r) := \arg\max_{t \in T}^{n}(|Y_t|) \tag{2.1}$$

2. Die *populärsten Tags einer Ressource* liefern die Tags, die am häufigsten von Nutzern für diese Ressource verwendet wurden:

$$\widetilde{T}(u, r) := \arg\max_{t \in T}^{n}(|Y_{t,r}|) \tag{2.2}$$

3. *Populärste Tags eines Benutzers* sind analog dazu nicht abhängig von der Ressource, aber vom Benutzer. So lassen sich die n-häufigsten Tags eines bestimmten Benutzers wie folgt bestimmen:

$$\widetilde{T}(u, r) := \arg\max_{t \in T}^{n}(|Y_{u,t}|) \tag{2.3}$$

Diese Tag-Recommender lassen sich gut miteinander kombinieren, in dem man deren Häufigkeiten addiert und die größten davon einsammelt:

$$\widetilde{T}(u, r) := \arg\max_{t \in T}^{n}(|Y_{t,r}| + |Y_{u,t}|) \tag{2.4}$$

Problematisch ist hierbei der sich mindernde Einfluss der nutzerbasierten Empfehlungen gegenüber den ressourcenbasierten Empfehlungen, wenn eine wachsende Nutzeranzahl diese Ressource verschlagwortet. Umgekehrt gilt gleiches. Abhilfe schafft hier eine Normalisierung der Werte im Intervall zwischen Null und Eins. Wenn sowohl für den nutzerbasierten als auch für den ressourcenbasierten Recommender der beste Tag den Wert Eins aufweist, spielt die Tatsache, dass viele Nutzer diese Ressource verschlagwortet haben, keine Rolle mehr.

2.3.2 Inhaltsbasierte Recommender

Recommender, die häufig genutzte Tags vorschlagen, haben das Problem des *cold start*. Bei einem neuen Benutzer, der bisher keine Ressourcen getagged hat, kann auch nicht auf sein Vokabular der Terme zurückgegriffen werden, da noch keins existiert. Sich allein auf die *populärsten Tags der Foksonomy* zu berufen ist nicht ausreichend, da die Verteilung der Terme bei großen Folksonomien wie Delicious den Gesetzmäßigkeiten des Potenzgesetzes folgt [CLP06]. Demnach ist zwar die Wahrscheinlichkeit hoch, dass die n-häufigst verwendeten Terme der Folksonomy im Vergleich zu den übrigen Termen passen könnten, aber es müssen nicht zwangsläufig die geeignetsten sein. Dieses Problem lässt sich lösen, wenn die Ressource herangezogen wird, um aus ihr selbst oder aus ihren Metadaten geeignete Terme zu extrahieren. Bei einer aus Text bestehenden Ressource wäre es ein naiver Ansatz den Text in Terme zu zerlegen und die Terme nach deren Häufigkeiten zu sortieren. Problematisch hierbei ist, dass die vorgeschlagenen Terme aufgrund ihres häufigen Auftretens vermutlich Präpositionen, Artikel oder Füllwörter sind. Diese Wörter sind als Tags nicht sonderlich gut geeignet, da sie für sich stehend wenig Bedeutung haben und nur im Kontext mit Nomen und Verben verwendet werden. Diese Stoppwörter müssen also zuvor herausgefiltert werden.

Eine weitere Möglichkeit zur Gewichtung extrahierter Terme ist die Häufigkeit der Nutzung dieser als Tag. Dieses Maß lässt sich normalisieren, indem durch das Maximum der erhaltenen Werte geteilt wird. Bei dieser Methode dürften die Stoppwörter automatisch herausgefiltert werden, da davon auszugehen ist, dass in den meisten Folksonomien Stoppwörter nicht als Tags verwendet wurden. Diese Methode wird aber nur bereits als Tags verwendete Terme vorschlagen.

Ein gutes Maß zur Gewichtung von Termen ist das *tf-idf-Maß* [BYRN99], das sich wie folgt bestimmt: Die *Vorkommenshäufigkeit* $tf_{i,j}$ (term frequency) gibt an, wie häufig der Term i im Dokument j vorkommt. Die inverse *Dokumentenhäufigkeit* idf_i (inverted document frequency) misst die allgemeine Bedeutung des Terms i für die Gesamtmenge der betrachteten Dokumente D. Hierfür wird die Gesamtanzahl der Dokumente $N = |D|$ durch die Anzahl der Dokumente, in denen der Term i vorkommt – hier n_i – geteilt. Um den starken Anstieg zu dämpfen, wird die Dokumentenhäufigkeit logarithmiert. Das

Produkt von *tf* und *idf* wird *tf-idf-Maß* genannt.

$$w_{i,j} = tf_{i,j} \cdot idf_i = tf_{i,j} \cdot \log \frac{N}{n_i} \tag{2.5}$$

Auf das Szenario eines Tag-Recommenders angewandt, erhält beispielsweise eine Präposition, die in der betrachteten Ressource häufig vorkommt, durch die tf-Funktion zunächst einen hohen Wert und durch die *idf*-Funktion einen niedrigen, da sie vermutlich auch in anderen Ressourcen häufig verwendet wurde. Wenn diese Präposition in jedem Dokument vorkommt, ist das Gewicht sogar Null, da $\log(1) = 0$. Die *idf*-Funktion wirkt als Korrektiv. Sie gewichtet in verhältnismäßig vielen Dokumenten vorkommende Terme weniger stark als solche, die in in wenigen Dokumenten vorkommen. Im Information Retrieval hat sich *tf-idf-Maß* zur Gewichtung von Termen bewährt. Von Nachteil ist der hohe Aufwand zur Bestimmung dieser Werte. Es bietet sich an einen Index anzulegen, der Methoden bereitstellt, mit denen die Termhäufigkeit und die Dokumentenhäufigkeit eines Terms ermittelt werden können.

Bei allen drei Methoden wird die Liste der Terme absteigend nach ihren Gewichten sortiert. Die ersten n Terme sind die Empfehlungen.

2.3.3 Kollaboratives Filtern

Aufgrund der Einfachheit und den vielversprechenden Ergebnissen findet Kollaboratives Filtern häufig Anwendung in Recommender-Systemen. Vorgeschlagen werden Objekte von gleichgesinnten Nutzern. Gleichgesinnte Benutzer sind diejenigen, die gleiche Objekte ähnlich bewertet haben. Eingesetzt wird dies beispielsweise im Online-Handel, wo die Nutzer Produkte bewerten können, mit deren Hilfe einem Kunden für ihn potentiell interessante Produkte vorgeschlagen werden. Ebenso findet diese Technik Anwendung in Sozialen Netzwerken, wie beispielsweise dem Internetradio *last.fm*,[4] das vielfältige Interaktionsmöglichkeiten der Benutzer bereithält. Hier werden mit Hilfe des Kollaborativen Filterns die Musikprofile der verschiedenen Nutzer miteinander verglichen und die Songs der ähnlichsten Nutzer verwendet, um dynamische Wiedergabelisten zu erstellen.

Die Funktionsweise des Kollaborativen Filterns basiert auf Bewertungen der Nutzer. Diese können als Benutzer-Objekt-Matrix dargestellt werden. Wird die Matrix in m Spaltenvektoren zerlegt, stellt jeder dieser Vektoren die Bewertungen der Objekte 1 bis n eines Benutzers dar:

$$\mathbf{X} := [\vec{x}_1, \ldots, \vec{x}_m]^T \text{ mit } \vec{x}_u := [x_{u,1}, \ldots, x_{u,n}] \text{ für } u := 1, \ldots, m,$$

wobei $x_{u,o}$ zeigt, dass der Nutzer u das Objekt o mit $x_{u,o} \in \mathbb{R}$ bewertet hat [Jäs11].

[4] `http://www.last.fm`

Die Menge der k-ähnlichsten Benutzer eines Benutzers u (k-Nachbarschaft N_u^k) lassen sich dann durch

$$N_u^k := \underset{v \in U \setminus \{u\}}{\arg\overset{k}{\max}} \operatorname{sim}(\vec{x}_u, \vec{x}_v) \tag{2.6}$$

berechnen.[5]

Um Kollaboratives Filtern in Folksonomien anwenden zu können, muss die ternäre Beziehung auf einen geringer dimensionierten Raum reduziert werden [TSMST08]. Hierfür können Projektionen genutzt werden, die entweder auf einer Benutzer-Ressourcen-Matrix oder auf einer Benutzer-Tag-Matrix abbilden. Diese Matrizen enthalten nur die Werte 0 (für Nicht-Auftreten) und 1 (für Auftreten).

1. Die Benutzer-Ressourcen-Matrix wird durch folgende Projektion erstellt:

$$\pi_{UR} Y \in \{0,1\}^{|\mathrm{U}|\times|\mathrm{R}|} \text{ mit } (\pi_{UR} Y)_{u,r} := \begin{cases} 1, & \text{falls } \exists t \in T, \text{ sodass } (u,r,t) \in Y \\ 0, & \text{sonst} \end{cases} \tag{2.7}$$

2. Analog dazu lautet die Projektionsvorschrift für die Benutzer-Tag-Matrix:

$$\pi_{UT} Y \in \{0,1\}^{|\mathrm{U}|\times|\mathrm{T}|} \text{ mit } (\pi_{UT} Y)_{u,t} := \begin{cases} 1, & \text{falls } \exists r \in R, \text{ sodass } (u,r,t) \in Y \\ 0, & \text{sonst} \end{cases} \tag{2.8}$$

Eine der beiden Möglichkeiten kann nun genutzt werden, um die k-Nachbarschaft des Benutzers u zu ermitteln, so wie es oben beschrieben ist. Die k-Nachbarschaft dient dann als Basis, um daraus Tags vorzuschlagen:

$$\widetilde{T}(u,r) := \underset{t \in T}{\arg\overset{n}{\max}} \sum_{v \in N_u^k} \operatorname{sim}(\vec{x}_u, \vec{x}_v)\, \delta(v,t,r) \tag{2.9}$$

wobei $\delta(v,t,r) := 1$ wenn $(v,t,r) \in Y$ und sonst 0 [Jäs11].

Auch bei dieser Methode gibt es das Problem des *cold start*. Hat der Benutzer u noch keine Ressourcen und Tags abgespeichert, kann auch keine Ähnlichkeit zu anderen Nutzern berechnet werden.

2.3.4 Kombination mehrerer Tag-Recommender

Die verschiedenen Verfahren haben ihre Vor- und Nachteile. Es ist daher naheliegend, mehrere Recommender zu entwickeln und diese miteinander zu kombinieren. Denn so

[5] Als Ähnlichkeitsfunktion für $\operatorname{sim}(\vec{x}_u, \vec{x}_v)$ kann beispielsweise das Kosinus-Ähnlichkeitsmaß oder der Pearson-Korrelationskoeffizient eingesetzt werden.

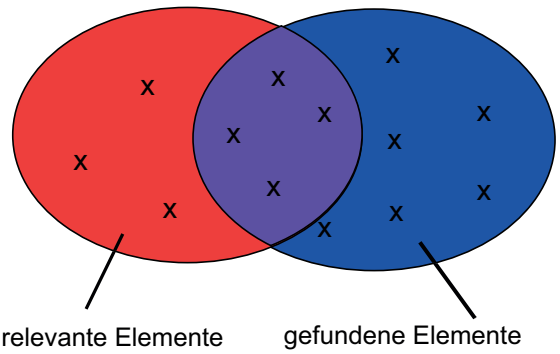


Abb. 2.3: Mengendarstellung relevanter und gefundener Elemente zur Bestimmung von Precision und Recall

wie sich die Recommender für die *populärsten Tags* kombinieren lassen, so lassen sich auch die Recommender der unterschiedlichen Verfahren miteinander zu einem hybriden Tag-Recommender verschmelzen, mit dem sich beispielsweise das Problem des *cold start* umgehen lässt. Im Kapitel 4 wird beschrieben, wie ein hybrider Tag-Recommender aufgebaut und praxistauglich umgesetzt werden kann.

2.4 Evaluation von Tag-Recommendern

Typischerweise werden im *Information Retrieval* Ergebnisse mit *Precision* (Genauigkeit) und *Recall* (Trefferquote) bewertet [SM83]. Das harmonische Mittel beider Werte wird *F-Maß* genannt. Um Precision und Recall bestimmen zu können, werden drei Werte benötigt:

- a = Anzahl gefundener und relevanter Ergebnisse der Suche,
- b = Anzahl gefundener und nicht-relevanter Ergebnisse der Suche und
- c = Anzahl nicht gefundener, aber relevanter Dokumente.

Mit diesen Werten lassen sich beide Maße berechnen: precision $= \frac{a}{a+b}$ und recall $= \frac{a}{a+c}$. Um *Precision* und *Recall* für die Evaluation eines Tag-Recommenders nutzen zu können, muss auf zwei zur Verfügung stehende Mengen zurückgegriffen werden. Das ist zum einen die Menge der Terme, die der Recommender vorschlägt – also $\widetilde{T}(u,r)$ – und zum anderen die Menge der Tags, die der Nutzer einer Ressource tatsächlich zuordnet – also $T(u,r)$. Der Wert a ist nichts weiter als die Menge der Elemente der Schnittmenge von $T(u,r)$ und $\widetilde{T}(u,r)$. Dementsprechend ist $a+b$ exakt die Anzahl der Elemente der Menge $\widetilde{T}(u,r)$ und analog entspricht $a+c$ der Anzahl der Elemente der Menge $T(u,r)$.

Precision und Recall lassen sich für Tag-Recommender demnach wie folgt bestimmen:

$$\text{precision}(\widetilde{T}(u,r)) = \frac{|T(u,r) \cap \widetilde{T}(u,r)|}{|\widetilde{T}(u,r)|} \tag{2.10}$$

$$\text{recall}(\widetilde{T}(u,r)) = \frac{|T(u,r) \cap \widetilde{T}(u,r)|}{|T(u,r)|}\,. \tag{2.11}$$

Beide Werte können als Wahrscheinlichkeitsmaße interpretiert werden. Precision ist demnach die Wahrscheinlichkeit, mit der ein gefundenes Element relevant ist und Recall die Wahrscheinlichkeit, mit der ein relevantes Element gefunden wird. Das F-Maß kombiniert Precision und Recall mittels des gewichteten harmonischen Mittels, wobei hier beide Werte gleich stark gewichtet werden:

$$F = \frac{2 \cdot \text{precision}(\widetilde{T}(u,r)) \cdot \text{recall}(\widetilde{T}(u,r))}{\text{precision}(\widetilde{T}(u,r)) + \text{recall}(\widetilde{T}(u,r))}\,. \tag{2.12}$$

2.5 Zusammenfassung

Als Folksonomy wird die Menge aller Nutzer, Ressourcen und Tags, sowie die Menge aller Zuordnungen zwischen diesen drei Mengen bezeichnet. Die daraus entstehende Struktur wird als Navigationselement in sozialer Software genutzt und ist daher nützlich für alle Benutzer dieser Software. Aus Nutzersicht gibt es individuelle und gruppenbezogene Motive für die Verwendung von Tags. Tag-Recommender sollen dem Nutzer Tags vorschlagen und ihm dadurch bei der kognitiv anspruchsvollen Arbeit des Indexierens unterstützen. Es wurden drei Verfahren für Tag-Empfehlungssysteme vorgestellt: Vorschlagen populärer Tags, inhaltsbasierte Tag-Generierung und Kollaboratives Filtern. Zur Bewertung von Tag-Recommendern werden die im Information Retrieval üblichen Maße Precision und Recall sowie das F-Maß, als harmonisches Mittel von Precision und Recall verwendet.

3 Architektur von UniVideo und Integration von Empfehlungsverfahren

Der für die Erweiterung um einen Tag-Recommender relevante Aufbau und die Funktionsweise von UniVideo wird im ersten Teil dieses Kapitels erläutert. Darauf aufbauend wird im zweiten Teil das entwickelte Framework für Tag-Recommender-Algorithmen vorgestellt.

3.1 Aufbau und Funktionsweise von UniVideo

UniVideo ist eine in PHP 5 entwickelte Webanwendung. Ursprünglich war es dazu gedacht, eine leicht zu bedienende Webschnittstelle zur Verfügung zu stellen, die es den Nutzern ermöglicht, Videos auf den Helix Video-Server des ITS zu laden, da zuvor der Upload für die meisten Benutzer zu umständlich über FTP und SSH bewerkstelligt werden musste.

Um jedes Video auf einer eigenen Seite mit Titel und Beschreibung präsentieren zu können und um auch diese Metadaten der Videos speichern zu können, wurde eine MySQL-Datenbank verwendet; Kommentar- und Taggingfunktion folgten. Auch aufgrund sich ändernder Rahmenbedingungen in der Serverinfrastruktur des ITS musste UniVideo mehrfach umgebaut werden. Durch diese stetige Weiterentwicklung wurde auch die Architektur fortlaufend angepasst. In seiner jetzigen Form basiert UniVideo auf dem Entwurfsmuster *Model-View-Controller*.

3.1.1 Model-View-Controller

Model-View-Controller (kurz MVC) ist ein in der Software-Entwicklung häufig verwendetes Architekturmodell (Design Pattern/Entwurfsmuster) zur Entwicklung von Software. Es besteht aus den drei Einheiten Model, View und Controller. Das Model (Datenmodell) enthält dabei nur die darzustellenden Daten. Der View (oder die Sicht) bewerkstelligt die Darstellung der Daten und der Controller (die Steuerung) verwaltet die Benutzeraktionen und steuert entsprechend die Sichten.

Das Datenmodell besteht in UniVideo aus den Entitäten *Video*, *User*, *Comment* und *Term*, welche alle von der abstrakten Klasse *AbstractEntity* erben. Diese Klasse ist jedoch

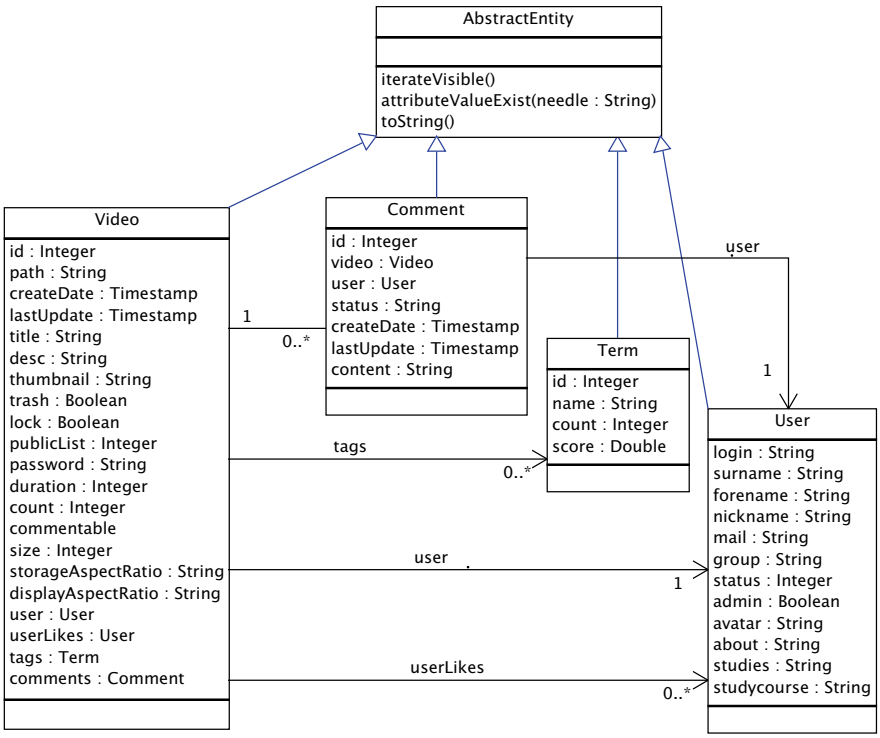


Abb. 3.1: Klassendiagramm des Datenmodells von UniVideo

nur ein Hilfskonstrukt, welches von anderen Teilen von UniVideo benötigt wird und für die Repräsentation des Datenmodells irrelevant ist. Abbildung 3.1 zeigt das Klassendiagramm zum Datenmodell. Auffällig ist, dass die meisten Assoziationen unidirektional sind. Auch wenn die jeweiligen Rückverbindungen im Klassendiagramm nicht ersichtlich sind, kann auf die Daten dennoch über die *Getter*-Methoden zugegriffen werden. Die *Getter*-Methoden laden im Bedarfsfall die Daten aus der Datenbank nach. Hier ist die Idee des *Lazy Loadings* umgesetzt. Die gelieferten Objekte enthalten noch nicht alle Daten, wissen aber wie sie diese beschaffen können und laden sie (nach), wenn auf die entsprechende *Getter*-Methode zugegriffen wird. Diese Strategie wurde gewählt, da für einige Sichten viele *Video*-Objekte geladen werden müssen. Wenn alle Video-Objekte die *User*-Objekte ihrer Besitzer enthalten und diese *User*-Objekte wiederum alle *Comment*-Objekte und zugehörigen *Video*-Objekte enthalten, so ist die Initialisierung einer solchen Sicht sehr teuer, obwohl der Großteil der geladenen Informationen gar nicht benötigt wird.

Der Controller hat die Aufgabe, die URLs zu parsen und entscheidet anhand dieser, welche Sicht der Betrachter angefordert hat. Er lädt den passenden View (Sicht) und bindet das dazugehörige Template (s. u.) und dessen Helferklassen ein. Abbildung 3.2 zeigt beispielhaft vier URLs und die dazugehörigen Sichten.

/ Startseite

/video/<id> zeigt View, für Video der ID <id>

/user/<login> zeigt View, welcher alle Videos des Benutzers <login> auflistet

/tag/<term> zeigt View, welcher alle Videos auflistet, die mit dem Term <term> verschlagwortet sind

/search/<string> zeigt View, welcher alle Ergebnisse der Suche nach <string> auflistet.

Abb. 3.2: URL-Schema der Sichten des Frontends in UniVideo

Der View lädt, noch bevor der Code des Templates ausgeführt wird, mit Hilfe von Datenbank-Klassen die Informationen der angeforderten Videos aus der Datenbank. Daraufhin werden Video-Objekte generiert. Diese werden in einer universellen Listenstruktur, welche alle Objekte vom Typ *AbstractEntity* enthalten kann, abgelegt. Über das *ViewHelper*-Objekt kann im Template auf diese Datenstruktur zugegriffen und iteriert werden. Das Template besteht aus HTML-Code und kurzen PHP-Tags. Die PHP-Tags dienen nur dazu, Methoden des ViewHelpers auszuführen, welche für die Iteration über die Listenstrukuren benötigt werden und auf die Attribute der Objekte zugreifen können.

In Listing 3.1 sehen wir ein Beispiel eines Templates, das die Videos eines bestimmten Tags mit einem kleinen Vorschaubild auflisten soll. Die `while`-Schleife bedient sich der Methode `has_more()` (Zeile 4) und prüft, ob in der Liste weitere Objekte enthalten sind. Wenn das der Fall ist, setzt die Methode `load_object()` den Zeiger auf dieses Objekt. Alle weiteren Methoden der *ViewHelper*-Klasse sind dafür gedacht die Programmlogik vom Template zu kapseln und machen nichts weiter, als die Attribute des Objekts auf das der Listenzeiger gerade zeigt zurückzugeben. Bietet der *ViewHelper* keine passende Methode an, kann man dennoch auf die Methoden des Objekts zugreifen, wie dies in Zeile 17 bespielsweise getan wird. Hier soll ausgegeben werden wie oft das Video schon angeschaut wurde. Da der *ViewHelper* keine entsprechende Methode anbietet, wird hier über `the_object()` das ganze Objekt geholt und über dessen Methode `getCount()` der gewünschte Wert beschafft.

Listing 3.1: Beispiel eines Tag-Templates in UniVideo

```
1  <div id="page">
2    <?php if($view->has_objects()) : ?>
3      <h1>Videos mit Tag: <?php echo $view->pagetype_value(); ?></h1>
4      <?php while ( $view->has_more() ) : $view->load_object(); ?>
5      <div class="video">
6        <?php echo $view->the_preview('128x72') ?>
7        <h4>
8          <a href="<?php echo $view->the_url(); ?>">
9            <?php echo $view->the_title(32); ?>
```

```
10          </a>
11        </h4>
12        <p>von
13          <a href="<?php echo
                Functions::uvid_getUrl('user',$view->the_user()->getLogin());
                ?>">
14            <?php echo $view->the_user()->getName(); ?>
15          </a>
16          <br />
17          <?php echo $view->the_object()->getCount(); ?> mal angesehen.
18        </p>
19      </div>
20      <?php endwhile; ?>
21    <?php else: ?>
22    <h1>Keine Videos</h1>
23    <?php endif; ?>
24    <p>
25      <?php echo $view->prev_page_link('Vorherige Seite');  ?>
26      <?php echo $view->next_page_link('Nächste Seite', 12);  ?>
27    </p>
28  </div>
```

3.1.2 Datenbankzugriff

Die Schnittstelle zur Datenbank ist über die *Datenbank-Manager-Klassen* realisiert, die für eine MySQL-Datenbank ausgelegt sind. Für jede Entität gibt es eine eigene Manager-Klasse, die Methoden bereitstellt, welche die gewünschten Entitäten (Video, Term, User, Comment) zurückgibt. Dazu werden in diesen Methoden die *SQL*-Abfragen mit dem jeweiligen *QueryBuilder* erstellt. Es gibt einen *SelectBuilder*, *InsertBuilder*, *UpdateBuilder* und *DeleteBuilder*. Über den Konstruktor oder andere Methoden wird festgelegt, welche Datensätze geladen, geändert, eingefügt oder gelöscht werden sollen. Über die `toString()`-Methode wird die SQL-Abfrage als Zeichenkette zurückgegeben. Diese Zeichenkette wird an die *Database*-Klasse zur Ausführung weitergereicht. Mit Hilfe der MySQL-Methoden von PHP,[1] welche als Schnittstelle zu einer MySQL-Datenbank dienen, wird der *SQL*-Code an die Datenbank übergeben und dort ausgeführt. Diese Methoden liefern bei einer Abfrage (*SELECT*) ein Array, das so genannte Resultset, zurück. Bei Manipulationsoperationen (*INSERT*, *UPDATE*, *DELETE*) wird die Operation bei Erfolg mit einem *TRUE* oder mit einem *FALSE* bestätigt. Bei einer Abfrage erzeugt der jeweilige Datenbank-Manager nun aus dem Resultset die gewünschten PHP-Objekte, die er dann zurückgibt. Sollte sich eine SQL-Abfrage mit Hilfe der *QueryBuilder* nicht

[1] Eine ausführliche Auflistung der MySQL-Schnittstelle in PHP findet sich unter `http://php.net/manual/en/book.mysql.php`

abbilden lassen, so besteht auch die Möglichkeit die SQL-Abfragen „per Hand“ zu erstellen und an die Datenbank zu reichen.

Alle Datenbank-Manager-Klassen sind mit dem Singleton-Pattern umgesetzt und können an jeder beliebigen Stelle des Programmcodes genutzt werden. Natürlich sollten sie aufgrund der Programmcode-Kapselung nicht im Template eingesetzt werden.

3.1.3 Abbildung der Tags in der Datenbank

Wie in Definition 2.1 dargestellt, ist eine Tag-Zuordnung die Relation zwischen jeweils einem Element aus drei unterschiedlichen Mengen. In der Datenbank von UniVideo ist dies, neben den Tabellen *videos* und *users*, durch die Tabellen *terms* und *term_rel* realisiert. In *terms* werden alle Terme mit der Beschränkung *UNIQUE* gespeichert. Diese Beschränkung legt fest, dass jeder Term nur einmal existieren darf. Jeder Term erhält auf diese Weise eine eindeutige ID. In *term_rel* wird diese ID, sowie die ID der Videos und der Benutzer als Fremdschlüssel gespeichert. Eine weitere Tabellenspalte *object_type* dient zur Unterscheidung, um welche Art von Term-Relation es sich handelt (vgl. Abb. 3.3).

	#	Spalte	Typ	Kollation
☐	1	**object_type**	varchar(255)	latin1_swedish_ci
☐	2	**object**	int(11)	
☐	3	**user**	varchar(20)	latin1_swedish_ci
☐	4	**term**	int(11)	
☐	5	**timestamp**	timestamp	

Abb. 3.3: Struktur der Tabelle *term_rel* in UniVideo

Anfangs war für UniVideo angedacht, dass neben dem Tagging zusätzlich Kategorien eingeführt werden. In diesem Fall hätte mit Hilfe von *object_type* eine entsprechende Unterscheidung gemacht werden können und so hätten auch Kategorien in *term_rel* abgebildet werden können. Die Fremdschlüssel-Spalten bilden zusammen mit *object_type* den Primary-Key der Tabelle *term_rel*. Damit wird sichergestellt, dass eine Tag-Zuordnung nicht mehrfach gespeichert werden kann. Die fünfte Spalte *timestamp* speichert beim Anlegen eines neuen Tags die aktuelle Zeit. Damit kann nachvollzogen werden, wann eine Tag-Zuordnung erstellt worden ist.

3.1.4 REST Web Services für AJAX-Requests

AJAX (Asynchronous JavaScript and XML) nennt man die Technik mit der eine asynchrone Datenübertragung zwischen Client (Browser) und Server ermöglicht wird, ohne dass dafür die Webseite neu geladen werden muss. Hierfür bietet JavaScript eine entsprechende Schnittstelle an. In UniVideo wird diese Technik eingesetzt, damit bei der Übertragung von Formularen nicht die jeweilige Sicht verlassen werden muss. Damit kann eine einzige

Sicht mehrere Zustände annehmen. Einige REST (Representational State Transfer) Web Services dienen als Schnittstelle zwischen Browser und Server, die die AJAX-Anfragen entgegennehmen. Die Idee von REST ist es, dass die angeforderte Ressource über eine eindeutige URL erreichbar ist. Der Web Service wertet die URL aus und schickt das angeforderte Objekt zurück. Der Client nimmt es entgegen und wechselt in einen neuen Zustand.

Die übermittelten Objekte müssen nicht zwangsläufig im XML-Format sein, obwohl der Name AJAX gegenteiliges fälschlicherweise suggeriert. UniVideo nutzt statt XML das kompaktere JSON[2]-Format. UniVideo bietet einige wenige Web Services an. So ist es beispielsweise möglich über diese Schnittstellen Video-Objekte anzufordern, Kommentare zu speichern und Benutzereinstellungen zu ändern. Die Web Services werden über URLs der Form `/api/<object_type>/<funktion>` angesprochen und die Parameter werden in der POST-Variable mitgegeben. Also `/api/video/get` oder `/api/video/update` wären zwei Beispiele, um einen Web Service anzusprechen. Der Aufbau der Web Services in UniVideo ist nicht konsistent im Sinne des Architekturstils von REST. Außerdem sind nicht alle Funktionen, die von einer solchen Schnittstelle erwartet werden, implementiert. Dennoch erfüllen sie ihren (bisherigen) Zweck. Zukünftig muss dieser Bereich sicherlich noch erweitert werden.

Auch die Tag-Empfehlungen sollten über AJAX in die bestehende Seite eingebunden werden. Das vereinfacht die Handhabung, da der Nutzer die Sicht nicht wechseln muss, um die Empfehlungen seines Videos zu erhalten. Um dies zu realisieren ist ein weiterer Web Service entwickelt worden, der diese Empfehlungen liefert.

3.2 Framework für Algorithmen und Evaluation

Zur Umsetzung und Evaluation der Algorithmen des Tag-Recommenders dient ein eigens dafür entwickeltes Framework, das sich in UniVideo einpasst und nötige Methoden und Schnittstellen zur Datenbank zur Verfügung stellt.

3.2.1 Aufbau und Integration des Frameworks in UniVideo

Das Klassendiagramm in Abbildung 3.4 zeigt den Aufbau des Frameworks und die Schnittstellen zu bestehenden Programmteilen von UniVideo. Wie zu sehen ist werden die Algorithmen in jeweils einzelnen Klassen beschrieben, die von der Klasse *AbstractAlgo* erben und die Methode `load_recommended_terms_for_object()` implementieren müssen. In dieser Methode wird der eigentliche Algorithmus umgesetzt. Über den Konstruktor werden das Video, für das die Tag-Empfehlungen berechnet werden sollen, und

[2]Der einfache Aufbau von JSON wird anschaulich unter `http://www.json.org` erklärt.

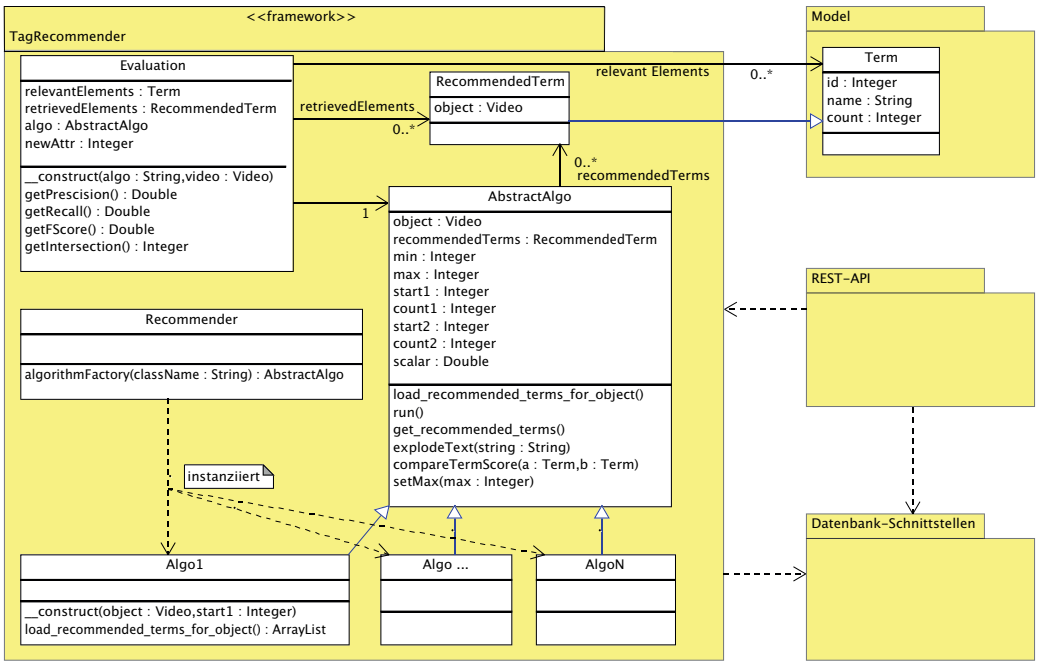


Abb. 3.4: Klassendiagramm des Recommender-Frameworks

weitere Parameter gesetzt, mit denen beschrieben wird, welche Daten als Grundlage zur Berechnung der Tag-Vorschläge herangezogen werden sollen. Dies ist wichtig, da für die Evaluation die Kreuzvalidierung zum Einsatz kommt und jeweils die Testdatenmenge von der Trainingsdatenmenge ausgespart werden muss (vgl. Abschnitt 5.1.2). Im realen Betrieb wird dies umgangen, indem alle vorhandenen Daten als Berechnungsgrundlage herangezogen werden. Die `run()`-Methode startet schließlich den Algorithmus. Über die Methode `get_recommended_terms()` wird nach der Berechnung auf die empfohlenen Terme zugegriffen.

Weiterer Bestandteil des Frameworks ist die Factory-Klasse *Recommender*. Sie enthält nur die Methode `algorithmFactory()`. Diese Methode erwartet den Klassennamen des gewünschten Algorithmus als String und instanziiert die gewünschte Klasse und liefert das Objekt zurück. Dafür bindet sie zunächst die benötigte PHP-Datei ein. Die Klasse *RecommendedTerm* erbt von der Klasse *Term* des Datenmodells. Im Gegensatz zu ihrer Mutterklasse, hat sie eine Assoziation zu dem zugehörigen Video-Objekt. Tag-Empfehlungen werden vom Framework immer als Liste von *RecommendedTerms* ausgeliefert.

Für die zur Evaluation notwendigen drei Maße, stellt die Klasse *Evaluation* entsprechende Methoden zur Berechnung zur Verfügung. Hierfür wird wieder das Video und der gewünschte Algorithmus benötigt. Der Konstruktor der Klasse instanziiert mit Hilfe der

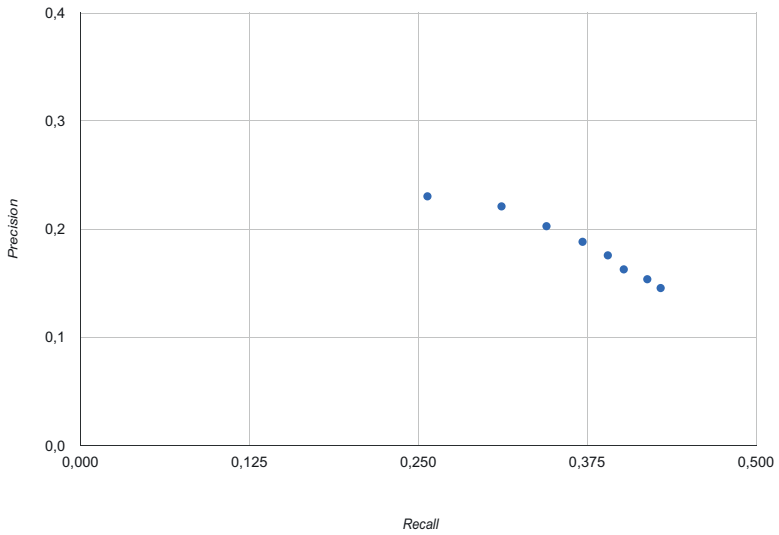


Abb. 3.5: Precision- und Recall-Darstellung als Punktdiagramm mit Google Chart Tools

Recommender-Klasse den Algorithmus.

Um Precision und Recall berechnen zu können, werden – wie in Abschnitt 2.4 beschrieben – die vom Benutzer tatsächlich verwendeten und über das Video-Objekt beschafften Tags $T(u, r)$, sowie die empfohlenen Tags $\tilde{T}(u, r)$ des Recommender-Algorithmus benötigt. Letztere liefert das Algorithmus-Objekt nach der Berechnung der Empfehlungen. Die Methode `getIntersection()` liefert die zur Berechnung benötigte Anzahl der Elemente der Schnittmenge der tatsächlich verwendeten Tags und der empfohlenen Tags (vgl. Abb. 2.3) zurück. Dafür zählt die Methode lediglich die gemeinsamen Elemente.

Die Methode `getFScore()` greift auf Precision und Recall zurück, um das F-Maß als harmonisches Mittel beider Werte – wie ebenfalls in Abschnitt 2.4 gezeigt – berechnen zu können.

3.2.2 Graphische Darstellung der Evaluation

Das Framework ist u. a. dafür gedacht, umgesetzte Algorithmen leicht testen und vergleichen zu können. Um die Vergleichbarkeit zu erhöhen, ist es von Vorteil die Ergebnisse in einem Diagramm darzustellen. Da UniVideo eine Webanwendung ist und um die Ergebnisse *live* sichtbar zu machen, wird ein Diagramm der Ergebnisse direkt nach der Berechnung als Website dargestellt. Um dies zu realisieren, wird die JavaScript-Bibliothek der *Google Chart Tools*[3] verwendet. Diese Bibliothek stellt die unterschiedlichsten Diagrammtypen

[3] `http://code.google.com/apis/chart/`

zur Verfügung, so auch *Scatter Charts* (Punktdiagramme). In Abbildung 3.5 wird ein Beispiel eines Punktdiagramms, das mit Hilfe von *Google Chart Tools* erstellt worden ist, gezeigt. Es zeigt das Verhalten eines Algorithmus bei unterschiedlicher Anzahl an Tag-Empfehlungen. An der X-Achse wird der Recall und an der Y-Achse die Precision gemessen. Mit steigender Anzahl von Tag-Empfehlungen steigt auch der Recall und in der Regel fällt mit steigendem Recall die Precision.

3.3 Benutzer-Schnittstellen für Upload und Tag-Recommender

Die Tag-Empfehlungen sollen dem Benutzer beim Hochladen der Videos präsentiert werden. Bisher war die Benutzerschnittstelle so realisiert, dass der Benutzer in einem Formular über die Eingabefelder Titel, Beschreibung und Tags eingibt und über ein *Input-File*-Feld die gewünschte Datei von seiner Festplatte auswählen konnte. Mit einem Klick auf Upload wurde dann das Video hochgeladen. Anschließend wurden Titel, Beschreibung und Tags in der Datenbank gespeichert sowie die Speicherung von drei Standbildern aus dem Video und die Konvertierung des Videos in das FLV-Format gestartet.

3.3.1 Upload-Formular

Eine Idee für einen Recommender ist es, mit Hilfe von OCR-Software (*Optical Character Recognition*) Standbilder des Videos nach Text zu durchsuchen und diesen Text für die Generierung von Tags zu nutzen. Dafür muss allerdings serverseitig das Video selbst zur Verfügung stehen. Der Upload musste also so umgebaut werden, dass zunächst nur das Video hochgeladen wird und dann die nötigen Skripte gestartet werden, welche Standbilder extrahieren und die OCR-Software starten. Anschließend soll der Benutzer zu einer anderen Seite weitergeleitet werden. Dort soll er aufgefordert werden Titel, Beschreibung und Tags einzugeben und außerdem sollen dort die Tag-Empfehlungen zur Auswahl stehen. Im gleichen Zuge sollte es ermöglicht werden, mehrere Videos gleichzeitig hochzuladen.

Für den Vorgang des Hochladens wird deswegen *Plupload*[4] genutzt. Mit dieser Sammlung von JavaScript-, PHP- und Flash[5]-Dateien ist nicht nur das Hochladen von mehreren Videos gleichzeitig möglich; sie ermöglichen außerdem eine Fortschrittsanzeige während des Hochladens (vgl. Abb. 3.6). Dies ist wichtig, da der Upload bei sehr großen Dateien – in UniVideo können Dateien bis zu einer Größe von 500 MB hochgeladen werden – sehr lange dauern kann. Wenn während dieser Zeit keine Rückmeldung vom System kommt, könnte das die Nutzer möglicherweise verunsichern. Die Nutzer könnten das

[4] Weitergehende Informationen und Dokumentation unter `http://www.plupload.com/`
[5] `http://www.adobe.com/de/flashplatform/`

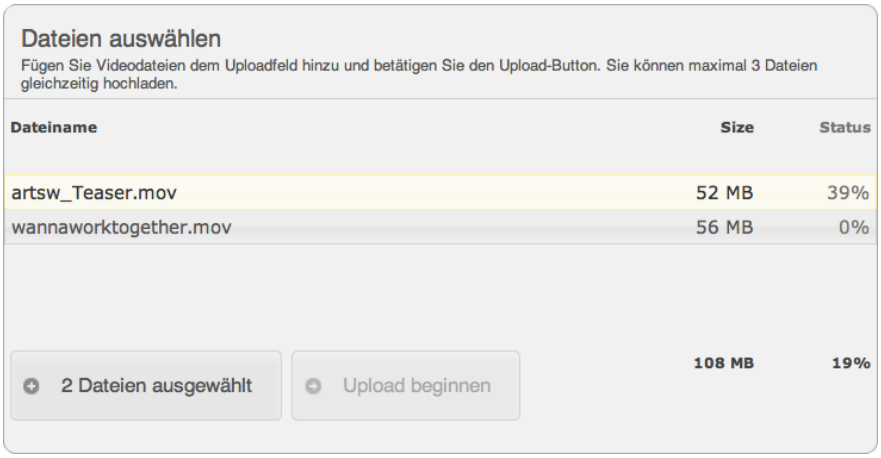


Abb. 3.6: Formular zum Hochladen von Videos in UniVideo

Browserfenster schließen, weil sie denken, dass aufgrund der Dauer der Vorgang vom Browser oder Server bereits abgebrochen wurde, obwohl das Video noch hochgeladen wird. Eine Statusmeldung über den Fortschritt des Hochladens, zeigt an, dass alles in Ordnung ist. Außerdem lässt sich *Plupload* so konfigurieren, dass es Dateitypen und Dateigröße prüft und ggf. eine Fehlermeldung ausgibt, bevor das Video hochgeladen wird.

Plupload funktioniert so, dass clientseitig mit Hilfe einer Flash-Anwendung die gewünschte Datei in kleine Stücke, so genannte *Chunks*, zerlegt wird, welche Stück für Stück hochgeladen werden. Serverseitig nimmt eine PHP-Datei diese *Chunks* entgegen und schickt ein Signal an den Client zurück, der daraufhin das nächste Stück zum Webserver schickt. Hat das letzte Chunk den Server erreicht, werden die serverseitigen Bearbeitungsprozesse (Umwandeln der Videos ins FLV-Format, die Generierung der Standbilder usw.) angestoßen (genauere Beschreibung in Kapitel 4) und das Video wird in der Datenbank gespeichert. Dabei wird zunächst der Dateiname des Videos als Titel und Beschreibung verwendet. Zusätzlich wird das Attribut *UPLOADINPROGRESS* auf 1 gesetzt, das dem System sagt, dass das Video noch hochgeladen wird und für andere Nutzer noch nicht sichtbar sein darf. Sofern weitere Videos zum Hochladen ausgewählt worden sind, werden diese nun nach und nach auf die gleiche Weise hochgeladen.

3.3.2 Formular zur Eingabe der Metadaten, Tags und Einstellungen

Nachdem alle Videos hochgeladen worden sind, wird der Nutzer auf eine neue Seite weitergeleitet. Der dazugehörige View (uploadfinished) lädt alle Videos des Benutzers, die in der Datenbank mit *UPLOADINPROGRESS* gekennzeichnet sind und stellt diese jeweils in einem Formular dar. Dabei ist zunächst nur das erste Formular eingeblendet.

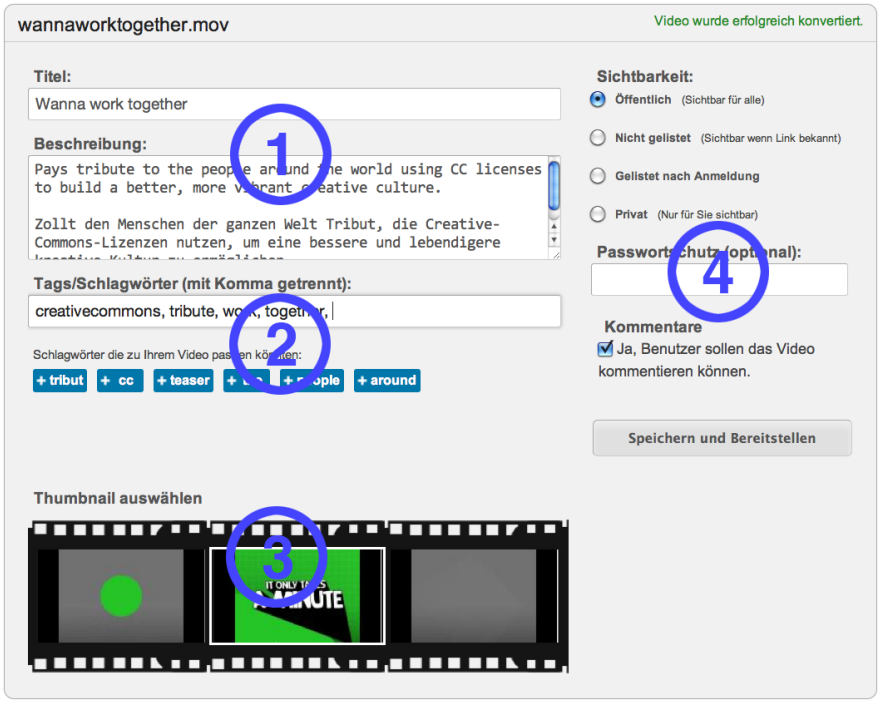


Abb. 3.7: Formular zur Eingabe der Metadaten, Tags und Einstellungen

Alle weiteren Formulare sind mittels CSS ausgeblendet (`display: none`) und werden sukzessive eingeblendet, sobald der Nutzer das Formular vollständig ausgefüllt und abgeschickt hat. Das Formular wird mit JavaScript gesteuert. Zunächst ist nur das Eingabefeld für den Titel für Eingaben aktiviert. Erst wenn der Nutzer hier etwas eingetragen hat, wird das Eingabefeld für die Beschreibung aktiviert. Ist auch eine Beschreibung hinzugefügt worden, wird das Eingabefeld für die Tags aktiviert und die Tag-Empfehlungen werden über einen AJAX-Request angefordert. Die strikte Reihenfolge der Eingabe ist notwendig, da der Recommender den Titel und die Beschreibung des Videos nutzen soll, um Empfehlungen zu generieren.

Wie in Abbildung 3.7 zu sehen ist, lässt sich das Eingabeformular in einen Kopf- und einen Hauptteil einteilen. Im Kopfteil wird links der Dateiname angezeigt, damit der Nutzer weiß, um welches Video es sich handelt. Rechts wird der Status der Konvertierung gezeigt und sekündlich aktualisiert. Der Hauptbereich besteht wiederum aus den vier folgenden Bereichen:

Bereich 1. Im ersten Bereich werden Titel und Beschreibung eingegeben. Sobald der Cursor eines der beiden Felder verlässt, werden über einen AJAX-Request die Formu-

lardaten an den Web Service `/api/video/upadate` übetragen und in der Datenbank gespeichert. Außerdem wird, sofern noch nicht geschehen, das nächste Eingabefeld aktiviert.

Bereich 2. Das Programm startet einen AJAX-Request an den Web Service `/api/tagrecommendation/get`, sobald der Cursor das Eingabefeld für die Beschreibung verlassen hat. Dieser startet nun serverseitig die Berechnung der Tag-Empfehlungen für dieses Video und schickt eine Liste von bis zu zehn Tag-Empfehlungen zurück. Wie in der Abbildung zu sehen ist, erscheinen diese Terme nun unter dem Eingabefeld für die Tags auf einem blauen Hintergrund mit einem weißen + davor. Das + soll symbolisieren, dass die blau unterlegten Terme anklickbar sind und auf diese Weise leicht in das Eingabefeld hinzugefügt werden können. Die Tags können aber genauso gut per Hand eingetragen werden. Hier wird ein Leerzeichen automatisch durch ein Komma ersetzt, da Kommata als Trennzeichen für die Tags genutzt werden. In der Abbilding nicht sichtbar sind noch zwei versteckte Felder (Hidden Fields[6]). Diese dienen dazu, die vorgeschlagenen Tags sowie die Tags, die der Nutzer durch anklicken ausgewählt hat zu speichern. Diese gespeicherten Tag-Empfehlungen werden für die Online-Evaluation benötigt (vgl. Abschnitt 5.3). Die Tag-Liste des Web Services wird als kommaseparierte Liste in das eine Hidden-Field eingefügt. Die durch den Nutzer angeklickten Tags werden kommasepariert in das andere Hidden-Field aufgenommen

Bereich 3. Im dritten Bereich kann eines von drei Standbildern als Thumbnail (also Vorschaubild) ausgewählt werden. Eine weiße Umrahmung deutet die Auswahl an.

Bereich 4. Der vierte Bereich wird aktiviert, sobald der erste und zweite Bereich ausgefüllt sind. Hier können Einstellungen zur Sichtbarkeit des Videos getätigt werden. Außerdem besteht die Möglichkeit das Video mit einem Passwort zu schützen. So kann der Nutzer sicherstellen, dass das Video nur denjenigen zugänglich ist, die über das Passwort verfügen. Weiterhin kann hier eingestellt werden, ob die Kommentarfunktion für dieses Video aktiviert werden soll. Der Button „Speichern und Bereitstellen“ übermittelt alle Daten des Formulars inklusive der Hidden-Fields an den Web Service `/api/uploadedvideo/update`. Dieser Web Service aktualisiert nochmals alle Formulareingaben und -einstellungen serverseitig, speichert die Tags und setzt das Attribut *UPLOADINPROGRESS* auf 0.

Sollte der Nutzer die gewünschten Felder nicht ausgefüllt haben, weil er das Browserfenster zuvor geschlossen hat oder in eine andere Sicht gewechselt ist, sind die hochgeladenen

[6] Vgl. auch `http://de.selfhtml.org/html/formulare/versteckte.htm`

Videos für ihn und andere Nutzer nicht sicht- und nutzbar. In diesem Fall ist weiterhin das Attribut *UPLOADINPROGRESS* auf 1 gesetzt. Das Video wird dadurch in keiner anderen Sicht erscheinen. Allerdings wird der Eigentümer darauf hingewiesen, dass noch Videos auf seine Bearbeitung warten. Über einen Link kann er jederzeit in die Sicht *uploadfinished* wechseln und die Beschriftung seiner hochgeladenen Videos abschließen und dadurch das Video bereitstellen.

3.4 Zusammenfassung

Es wurde erläutert wie die Rahmenbedingungen für einen im laufenden Betrieb befindlichen Tag-Recommender geschaffen worden sind. Dabei wurde gezeigt, wie sich ein Empfehlungssystem in die bestehende Struktur von UniVideo integrieren lässt. Bei der Umsetzung der Benutzerschnittstellen wurde ein hoher Wert auf die benutzerfreundliche Bedienung gelegt und zugleich sichergestellt, dass der Nutzer auch die nötigen Eingaben tätigt, die für die Berechnung von Tag-Empfehlungen notwendig sind. Im nun folgenden Kapitel wird der Fokus auf die Umsetzung der Algorithmen für Tag-Empfehlungen gelegt und die verschiedenen Strategien zur Berechnung dieser werden diskutiert.

4 Algorithmen für Tag-Empfehlungen in UniVideo

4.1 Vorüberlegungen

Ziel dieser Arbeit ist die Schaffung eines Tag-Empfehlungs-Systems, welches dem Benutzer für ihn nützliche Tags vorschlägt. In einer Folksonomy helfen die Tags zur Strukturierung der eigenen Inhalte und darüber hinaus entwickelt sich durch diese individuelle Strukturierung automatisch eine Ordnung, die eine kollaborative Dimension besitzt und demnach für alle Nutzer der Folksonomy Vorteile oder Nachteile bringen kann, da die entstehende Ordnung vom Gesamtsystem zur Navigation genutzt wird. Die Bedeutung des Taggens für den einzelnen Nutzer sowie für die gesamte Community, aus jeweils organisatorischen und kommunikativen Apekten, wurde in Abschnitt 2.2 diskutiert.[1] Zusammenfassend kann festgehalten werden, dass es vier Motive für das Taggen aus Nutzersicht gibt:

- persönliche Vorteile zur Organisation der eigenen Ressourcen (Klassifizieren der eigenen Inhalte, um diese schnell wiederzufinden)
- persönlich-kommunikative Vorteile (Dokumentation von eigenen Ideen und Erinnerungen)
- gruppenbezogene Vorteile zur Organisation (höhere Indexierungsebene, die der Gruppe das Finden von Ressourcen erleichtert; Vereinheitlichung des Vokabulars)
- gruppenbezogene Vorteile zur Kommunikation (eine Gruppe von Nutzern benutzen gemeinsame Tags, um auf ihre Inhalte aufmerksam zu machen oder Gruppenzugehörigkeit auszudrücken)

Idealerweise sollte der zu entwickelnde Tag-Recommender alle diese Aspekte bei der Generierung seiner Tags abdecken. Um dies zu erreichen, sind teilweise unterschiedliche Herangehensweisen für Algorithmen und unterschiedliche Quellen als Basis für Tag-Empfehlungen notwendig. Insgesamt wurden vier Algorithmen umgesetzt und getestet, die aus jeweils unterschiedlichen Quellen Terme zur Gewinnung von Tag-Empfehlungen heranziehen. Diese vier Algorithmen stützen sich auf die vorgestellten Verfahren aus Abschnitt 2.3 oder sind eine Weiterentwicklung dieser Ideen. Der fünfte Tag-Recommender ist ein

[1] Vgl. auch Abb. 2.2

hybrider Recommender, also eine Kombination aus den vier einzelnen Teil-Recommendern. In diesem Kapitel werden die Funktionsweisen dieser Recommender allgemein erklärt und es wird jeweils kurz auf die Umsetzung in UniVideo eingegangen. Die Bewertung der Teil-Recommender und des hybriden Recommenders folgt in Kapitel 5.

4.2 Populäre Tags (User Profile)

In Kapitel 2.3.1 wurden drei Möglichkeiten für die Umsetzung eines Tag-Recommenders auf der Basis populärer Tags gezeigt. Mit der Grundlage einer engen Folksonomy lassen sich nur zwei dieser Möglichkeiten in UniVideo umsetzen. Da Benutzer nur ihre eigenen Ressourcen (Videos) verschlagworten können, können gemeinsam verschlagwortete Ressourcen als Quelle für Tag-Empfehlungen nicht herangezogen werden. Der einfachste Ansatz wäre die meist verwendeten Tags der Folksonomy vorzuschlagen. Da dann jedem Benutzer unabhängig von seiner Ressource immer die gleichen Tags empfohlen werden, wird diese Strategie für die meisten Benutzer respektive Ressourcen nicht sonderlich gute Ergebnisse liefern.

Eine bessere Strategie ist es, die populärsten Tags des Benutzers vorzuschlagen, da davon ausgegangen werden kann, dass die meisten Nutzer häufig Videos eines bestimmten Themas (oder zu diesem verwandten Thema) in UniVideo zur Verfügung stellen.

$$\widetilde{T}(u,r) := \arg\max_{t \in T}^{n}(|Y_{u,t}|) \text{ mit } Y_{u,t} := Y \cap (\{u\} \times R \times \{t\})$$

Diese Strategie kann als *frequently used tags* bezeichnet werden (vgl. Kapitel 2.3.1). Eine zweite Annahme beruht auf der Beobachtung, dass Nutzer mehrere Videos eines bestimmten Themas oft in einem kurzen Zeitintervall hochladen. Als Beispiel hierfür seien die Video-Clips des Kasseler Science Slams genannt.[2] Zweimal fand dieses Veranstaltungsformat bisher in Kassel statt, das erste Mal im November 2010 und das zweite Mal im Mai 2011. Jeweils wenige Tage nach der Veranstaltung wurden die Videoaufzeichnungen in Form kurzer Clips in UniVideo bereitgestellt. Es erscheint naheliegend, dass dem Nutzer die kurz zuvor verwendeten Tags vorgeschlagen werden. Diese Strategie kann *recently used tags* genannt werden. Beide Ideen sind im Algorithmus 4.1 *User Profile* zusammengefasst.

Das Zeitintervall wird auf 24 Stunden festgelegt. Die Menge Y_{24h} wird als Menge aller Tag-Zuordnungen der vergangenen 24 Stunden definiert und $Y_u := Y \cap (\{u\} \times R \times T)$ (vgl. Kapitel 2.3.1) ist die Menge aller Tag-Zuordnungen des Benutzers u. Die Schnittmenge dieser beiden Mengen $Y_{u,24h}$ ist die Menge aller Tag-Zuordnungen des Benutzers u der vergangenen 24 Stunden. Wie in Algorithmus 4.1 zu sehen ist, werden alle Terme $t \in T$

[2] `http://univideo.uni-kassel.de/tag/scienceslam`

Algorithmus 4.1 User Profile

Require: Y, Y_u, Y_{24h} mit $Y_{24h} \subseteq Y$

1: **function** LOAD_RECOMMENDED_TERMS_FOR_OBJECT
2: $\widetilde{T} \leftarrow \{\}$
3: $T_{u,24h} \leftarrow \{t \mid (u,r,t) \in Y_{24h} \cap Y_u\}$
4: **if** $|T_{u,24h}| > 0$ **then**
5: **for all** $t \in T_{u,24h}$ **do**
6: $Y_{u,t,24h} \leftarrow \{(u,r,t) \mid (u,r,t) \in Y_{24h} \cap (\{u\} \times \{t\} \times R)\}$
7: $Y_{u,t} \leftarrow \{(u,r,t) \mid (u,r,t) \in Y_u \cap (U \times \{t\} \times R)\}$
8: $w \leftarrow \frac{|Y_{u,t,24h}|}{|Y_{u,t}|}$
9: $\tilde{T} \leftarrow \tilde{T} \cup \{(t,w)\}$
10: **end for**
11: **else**
12: $T_u \leftarrow \{t \mid (u,r,t) \in Y_u\}$
13: **for all** $t \in T_u$ **do**
14: $Y_{u,t} \leftarrow \{(u,r,t) \mid (u,r,t) \in Y_u \cap (U \times \{t\} \times R)\}$
15: $w \leftarrow \frac{|Y_{u,t}|}{|Y_u|}$
16: $\widetilde{T} \leftarrow \widetilde{T} \cup \{(t,w)\}$
17: **end for**
18: **end if**
19: **return** sort_descending($\widetilde{T}, \widetilde{t}_w$)
20: **end function**

aus $Y_{u,24h}$ gewählt (Zeile 3). Ist die Anzahl dieser Terme größer Null (Zeile 4), wird für jedes dieser Terme das Gewicht berechnet, indem die Anzahl der Tag-Zuordnungen des Benutzers u mit dem Term t der letzten 24 Stunden (Zeile 6) durch die Gesamtzahl der Tag-Zuordnungen dieses Benutzers u und dieses Terms t (Zeile 7) geteilt wird. Dieses Maß spiegelt das Verhältnis zwischen den in den letzten 24 Stunden am häufigsten verwendeten Tags des Nutzers und die Gesamtnutzung dieser Tags des Nutzers wider (Zeile 8) und entspricht der Idee der *recently used tags*. Hat der Benutzer ein bestimmtes Tag in den letzten 24 Stunden das erste Mal verwendet, dann bekommt das Gewicht dieses Tags den Maximalwert 1; benutzt er das Tag häufiger, ist das Gewicht entsprechend kleiner.

Sollte es keine Tag-Zuordnungen des Benutzers aus den vergangenen 24 Stunden geben (Zeile 11), werden alle seine Tags für die Berechnung der Term-Gewichte verwendet (Zeile 12). Das Gewicht für jedes seiner Tags (Zeile 13) wird berechnet, indem die Anzahl der Nutzung dieses Tags durch die Gesamtzahl seiner Tag-Zuordnungen geteilt wird (Zeile 15). Dabei bekommen seine häufiger verwendeten Tags ein höheres Gewicht als die weniger häufig verwendeten (*frequently used terms*).

Praktisch lässt sich dieser Algorithmus sehr leicht umsetzen, da mit Hilfe von einfachen Datenbankabfragen alle benötigten Mengen und Werte beschafft werden können. Dazu wird der Verbund aus U, R und T (In UniVideo heißen die Tabellen *users*, *videos* und *terms*) gebildet und nach den gewünschten Werten selektiert, also nach dem gewünschten Benutzer und/oder Tag und ggf. mit einer Beschränkung der Auswahl auf den Zeitraum der vergangenen 24 Stunden. Die Datenbank-Schnittstelle in UniVideo bietet die Funktion `numRows()`, die die Anzahl der selektierten Datensätze nach einer Datenbankabfrage zurückliefert. Nach der Berechnung der Gewichte werden abschließend die Tags nach diesen absteigend sortiert zurückgegeben.

4.3 Tag-Gewinnung aus Metadaten (Metadata)

Wie gezeigt werden konnte, kann eine Folksonomy eine gute Quelle für Tag-Empfehlungen sein. Lipczak und Milios haben untersucht wie gut außerdem der Titel einer Ressource als Quelle für Tag-Empfehlungen geeignet ist [LM10]. Sie bezeichnen den Titel als eine „verdichtete Ressourcen-Beschreibung“ die daher „wahrscheinlich nützliche Schlüsselwörter enthält“[3]. Dafür haben sie das Vorkommen von Termen als Tags und als Wörter im Titel untersucht und konnten bestätigen, dass die Nutzung eines Terms als Tag-Zuordnung von seiner Nutzung im Titel abhängt. Es liegt also nahe – zur Gewinnung von Tag-Empfehlungen für UniVideo – ebenso den Titel der Ressourcen zu verwenden. Da die vom Nutzer eingegebene Beschreibung ebenso nützliche Terme enthalten könnte, werden hier Titel *und* Beschreibung genutzt.

Für die Berechnung der Term-Gewichte wird angenommen, dass bereits als Tags genutzte Terme aus den Metadaten eines Videos, potentiell als Tags geeignet sind. Um nun die Termgewichte zu berechnen, muss für jeden Term die Anzahl der Videos ermittelt werden, in denen der Term in den Metadaten vorkommt. Im Folgenden wird dieser Wert mit a bezeichnet. Als zweiten Wert muss die Anzahl der Videos ermittelt werden, in denen der Term in den Metadaten vorkommt *und* als Tag verwendet worden ist – im Folgenden b genannt.

Das Gewicht w eines Terms $\tilde{t}$ lässt sich nun berechnen in dem b durch a geteilt wird:

$$w_{\hat{t}} = \frac{b}{a} \tag{4.1}$$

Da die Bedingung für b stärker ist als die für a, ist der Wert a immer größer als oder gleichgroß dem Wert von b. Infolgedessen kann das Gewicht des Terms niemals größer als 1 werden. Die Gewichte der Terme sind somit automatisch im Intervall zwischen 0 und 1 normalisiert.

[3] Im Original: The title, as a dense resource description, is likely to contain useful keywords.

Algorithmus 4.2 Tag-Gewinnung aus Metadaten

Require: Video r, Database db

1: **function** LOAD_RECOMMENDED_TERMS_FOR_OBJECT
2: $\widetilde{T} \leftarrow \{\}$
3: $\hat{T}_r \leftarrow \text{explodeText}(r_{title}, r_{decription})$
4: **for all** $\hat{t} \in \hat{T}_r$ **do**
5: $\hat{t} \leftarrow \text{toLowerCase}(\hat{t})$
6: **if** $\hat{t} \notin \widetilde{T}$ **then**
7: $w_{\hat{t}} \leftarrow 0$
8: $a \leftarrow db.\text{numOfTermOccurenceInMetadata}(\hat{t})$
9: **if** $a > 0$ **then**
10: $b \leftarrow db.\text{numOfTermOccurenceInMetadataAndTags}(\hat{t})$
11: $w_{\hat{t}} \leftarrow \frac{b}{a}$
12: **end if**
13: $\widetilde{T} \leftarrow \widetilde{T} \cup \{(\hat{t}, w_{\hat{t}})\}$
14: **end if**
15: **end for**
16: **return** sort_descending$(\widetilde{T}, \tilde{t}_w)$
17: **end function**

Bei der praktischen Anwendung in UniVideo lassen sich die Werte a und b wieder relativ einfach über entsprechende Datenbankabfragen ermitteln. Dazu muss wieder der Verbund (Join) zwischen den Tabellen *videos*, *terms* und *user* gebildet werden. Daraufhin werden für a die Datensätze selektiert und gezählt, in denen der Term in Titel oder Beschreibung vorkommt. Für den Wert b werden analog dazu die Datensätze gezählt, bei denen der Term in Titel oder Beschreibung *und* in den Tags vorkommt.

Algorithmus 4.2 verdeutlicht den Gesamtablauf. Der Titel und die Beschreibung des Videos werden hierfür in eine Menge einzelnener Terme zerlegt $\hat{T}_r$ (Zeile 3). Über eine Schleife wird über diese Terme iteriert (Zeile 4). Da in UniVideo alle Tags kleingeschrieben sind, werden zunächst im Schleifenrumpf alle Großbuchstaben des Terms in Kleinbuchstaben umgewandelt (Zeile 5). Anschließend wird geprüft, ob der jeweilige Term $\hat{t}$ bereits in der Ergebnismenge $\widetilde{T}$ vorhanden ist (Zeile 6). Wenn nicht, wird die Häufigkeit des Vorkommens in den Metadaten (a) mit Hilfe einer Datenbankabfrage in Erfahrung gebracht (Zeile 8). Anschließend wird die Vorkommenshäufigkeit in Metadaten *und* Tags (b) ermittelt (Zeile 10), um dann das Gewicht berechnen zu können (Zeile 11). Der Term mit seinem berechneten Gewicht wird in die Ergebnisliste aufgenommen. Nachdem der Schleifendurchlauf beendet ist, wird die Liste nach ihren Gewichten absteigend sortiert zurückgegeben. Die n am höchsten gewichteten Elemente sind die Tag-Empfehlungen.

4.4 Tag-Gewinnung mit OCR (OCR)

Ein Phänomen in Video-Portalen ist es, dass die Nutzer dazu geneigt sind, den Videos nur sehr knappe Beschreibungen hinzuzufügen. Dadurch entsteht ein Nachteil für den Recommender *Metadata*. Je weniger Text in den Metadaten vorhanden ist, umso weniger Tags können folglich daraus gewonnen werden. Als zweite Quelle zur Gewinnung von Tags, könnte das in Binärform vorliegende Video selbst dienen, indem Textpassagen des Videos mit einer OCR-Software (Optical Character Recognition) extrahiert werden. Das setzt natürlich voraus, dass das Video auch Textpassagen enthält. Da in UniVideo vielfach Vorlesungsaufzeichnungen hochgeladen werden, die die Folien der Präsentation des Dozierenden enthalten, gibt es entsprechendes Potential, das den Aufwand dieser Strategie rechtfertigt.

Der *OCR*-Tag-Recommender funktioniert auf eine dem *Metadata*-Recommender ähnliche Weise. Allerdings wird der *OCR*-Recommender in zwei zeitlich voneinander getrennten Schritten ausgeführt: Term-Gewinnung während des Hochladens des Videos und Term-Gewichtung während der Client die Empfehlungen anfordert.

Der erste Schritt geschieht also – für den Nutzer nicht sichtbar – unmittelbar nachdem das Video vollständig auf den Server geladen wurde. Für den Nutzer spürbar ist lediglich die Fortschrittsanzeige, die im Upload-Formular einige Sekunden (abhängig von der Größe des Videos) bei 100 Prozent verharrt, bevor das nächste Video hochgeladen wird bzw. der Nutzer zur nächsten Sicht (*uploadfinished*) geleitet wird (vgl. Kapitel 3.3). Während die Upload-Fortschrittsanzeige bei 100% verharrt, wird serverseitig das Video in 20 zeitlich gleichverteilte Standbilder zerlegt. Aus diesen 20 Bildern wird mit einer OCR-Software versucht Text zu extrahieren. Dieser Text wird in einzelne Wörter zerlegt. Da nicht davon ausgegangen werden kann, dass die OCR-Software immer korrekt geschriebene Wörter zurückliefert, wird jedes Wort einzeln auf seine Rechtschreibung geprüft. Nur Wörter, welche von der Rechtschreibprüfung erkannt werden, werden in die Liste der potentiell nützlichen Terme aufgenommen. In Kapitel 3.3.2 haben wir gesehen, dass die Tag-Empfehlungen dann abgerufen werden, wenn der Nutzer seinen Cursor in das Eingabefeld für die Tags setzt. Tests haben gezeigt, dass diese Art der Term-Gewinnung einige Sekunden pro Video benötigt (abhängig von der Länge des Videos und der Anzahl der gefundenen Terme). Die Reaktionszeit des Servers darf aber nicht allzu hoch sein, da der Nutzer nicht warten wird bis der Recommender die Empfehlungen liefert. Da insbesondere die Rechtschreibprüfung relativ viel Zeit in Anspruch nimmt, ist diese Aufteilung in zwei Schritte notwendig. Im zweiten Schritt werden dann schließlich die Terme berechnet.

In Algorithmus 4.3 wird der schematische Ablauf zusammenfassend skizziert. In diesem Beispiel kommt zur Erzeugung der Standbilder *mplayer*[4] zum Einsatz (vgl. Algorithmus

[4]mplayer steht für nahezu jedes Betriebssystem zur Verfügung und lässt sich über Shellkommandos

4.3 Zeile 1). Als Texterkennungswerkzeug dient *GOCR*.[5] Je höher der Kontrast zwischen Text und Hintergrund in einem Bild ist, umso besser funktioniert OCR. Um den Konstrast zu erhöhen, werden die Standbilder zunächst mit *djpeg*[6] von ihrem originalen Farbraum in einen graustufigen umgewandelt (Zeile 3). Für die Rechtschreibkorrektur kommt hier *aspell*[7] zum Einsatz. Zunächst war die Verwendung von Texterkennungswerkzeugen für einen inhaltsbasierten Tag-Recommender ein vielversprechender Ansatz, da bei Tests sehr gut brauchbare Terme extrahiert werden konnten. Im Anhang A.1 befinden sich Beispiel-Standbilder eines Videos mit Text-Passagen, aus denen Terme extrahiert werden konnten.

Algorithmus 4.3 Term-Extrahierung aus Videos mit Texterkennung

Require: $file$, *duration*, Database *db*, Video *video*

```
 1: cmd(mplayer file -vo jpeg -nosound -frames 20 -sstep duration/20)
 2: for all thumbs as thumb do
 3:     content ← cmd(djpeg -pnm -grayscale thumb | gocr - 2>&1)
 4:     terms ← explodeText(content)
 5:     for all terms as term do
 6:         if check_spelling(term) = 'ok' then
 7:             if term_video not in db_extracted_terms then
 8:                 db.insertExtractedTerm(video_id, term)
 9:             end if
10:         end if
11:     end for
12: end for
```

Der zweite Schritt besteht aus der Berechnung der Gewichte der gefundenen Terme, die auf ähnliche Weise geschieht, wie in Kapitel 4.3 beschrieben. Der Algorithmus zur Berechnung der Termgewichte ist identisch zum Algorithmus 4.2, lediglich die Quelle, aus der die Terme gewonnen werden, unterscheidet sich. $\hat{T}_r$ ist also die Menge der per OCR extrahierten Terme eines Videos r, reduziert um Stopp- und per Rechtschreibkorrektur herausgefilterte Wörter. Das Gewicht eines jeden Terms $\hat{t} \in \hat{T}_r$ bestimmt sich dann aus der Vorkommenshäufigkeit im Titel oder der Beschreibung *und* als Tag, geteilt durch die Vorkommenshäufigkeit im Titel oder der Beschreibung.

steuern (siehe `http://www.mplayerhq.hu/`).

[5] `http://jocr.sourceforge.net/`

[6] `http://www.masterscripts.de/manpage/d/djpeg.html`

[7] `http://aspell.net/`

4.5 Tags ähnlicher Videos (Similar Videos)

Die Idee hinter dem vierten Algorithmus ist es, dass für ein Video r die Tags ähnlicher Videos passen könnten. Dafür wird, ähnlich wie beim Kollaborativen Filtern (vgl. Kapitel 2.3.3) die k-Nachbarschaft eines Benutzers ermittelt wird, zunächst die k-Nachbarschaft der Ressource r ermittelt – also die k-ähnlichsten Videos von r:

$$N_r^k := \underset{\hat{r} \in R \setminus \{r\}}{\overset{k}{\arg\max}} \operatorname{sim}(r, \hat{r}). \tag{4.2}$$

Aus dieser Menge von Ressourcen dienen deren angehängte Tags als Grundlage für diesen Tag-Recommender:

$$\widetilde{T}(u, r) \subseteq \{t \mid (u, r, t) \in Y \cap \{U \times N_r^k \times T\}\}. \tag{4.3}$$

4.5.1 Berechnung der k-Nachbarschaft mit Lucene

Um die k-Nachbarschaft eines Videos zu ermitteln, kommt *Zend_Search_Lucene*[8] des *Zend Frameworks* zum Einsatz. Das *Zend Framework*[9] ist ein aus vielen Komponenten bestehendes Framework für PHP-Umgebungen. Es ist OpenSource-Software und steht unter der BSD-Lizenz.[10] Die einzelnen Komponenten dieses Projekts bieten Lösungen für typische Aufgaben von Web-Anwendungen auf PHP-Basis. Alle Komponenten sind unabhängig von einander nutzbar. *Zend_Search_Lucene* ist eine in PHP 5 geschriebene Textsuchmaschine und stellt eine dieser Komponenten dar. *Zend_Search_Lucene* ist ein Nachbau von *Apache Lucene*,[11] eine in Java entwickelte Programmbibliothek zur Volltextsuche. *Zend_Search_Lucene* unterstützt derzeit die Index-Formate der Versionen 1.4 bis 2.3 von *Apache Lucene*.

Um ähnliche Videos mit Hilfe einer Suchmaschine ermitteln zu können, müssen alle Videos im Index der Suchmaschine vorhanden sein. Jedes Video – respektive dessen Metadaten – ist also als atomares Dokument im Index gespeichert. Die *ID*, der Titel, die Beschreibung, die Tags und der Name des Besitzers des Videos werden als so genannte *Keyword*-Felder abgelegt, nach denen der Index durchsucht werden kann. Beim Hochladen eines Videos werden die genannten Informationen des Videos in den Index aufgenommen, so dass der Index immer aktuell ist.

Zend_Search_Lucene nutzt die gleichen Gewichtungsalgorithmen wie *Apache Lucene* auf Basis des Vektorraum-Modells. Bei einer Suchanfrage an den Index kommt das

[8] `http://framework.zend.com/manual/de/zend.search.lucene.overview.html`
[9] `http://framework.zend.com/`
[10] `http://www.opensource.org/licenses/BSD-3-Clause`
[11] `http://lucene.apache.org/`

Kosinus-Ähnlichkeitsmaß zum Einsatz, bei dem die Anfrage $\vec{q}$ und die Dokumente $\vec{d} \in D$ (D ist der Korpus des Index) die Vektoren ihrer Termgewichte sind:

$$\text{sim}(\vec{q}, \vec{d}) := \frac{\langle \vec{q}, \vec{d} \rangle}{\|\vec{q}\| \|\vec{d}\|}. \tag{4.4}$$

Die Termgewichte wiederum werden mit dem *tf-idf*-Maß (vgl. Kapitel 2.3.2) bestimmt. Die Suchergebnisse werden als eine Liste der gefundenen Dokumente mit dem zusätzlichen Attribut *Score* zurückgegeben. Dieser Wert ist die im Intervall zwischen 0 und 1 normalisierte Kosinus-Ähnlichkeit.[12]

Algorithmus 4.4 Tags ähnlicher Videos

Require: Y, R, Video r

function LOAD_RECOMMENDED_TERMS_FOR_OBJECT
  $\widetilde{T} \leftarrow \{\}$
  $N_{\hat{r}}^k := \underset{\hat{r} \in R \setminus \{r\}}{\arg\max^k} \text{sim}(r, \hat{r})$
  $w_{max} \leftarrow 0$
  for all $\hat{r} \in N_{\hat{r}}^k$ **do**
    $T_{\hat{r}} \leftarrow \{t \mid (u, r, t) \in Y \cap (U \times \{\hat{r}\} \times T)\}$
    for all $t \in T_{\hat{r}}$ **do**
      $tf \leftarrow \text{termOccurence}(t, r)$
      $df \leftarrow \text{docFreq}(t)$
      $w \leftarrow \hat{r}_{score} \cdot tf \cdot \log_{10} \frac{|R|}{1+df}$
      if $w_{max} < w$ **then**
        $w_{max} \leftarrow w$
      end if
      $\widetilde{T} \leftarrow \widetilde{T} \cup \{(t, w)\}$
    end for
  end for
  for all $w \in \{w \mid (t, w) \in \widetilde{T}\}$ **do**
    $w \leftarrow \frac{w}{w_{max}}$
  end for
  return sort_descending($\widetilde{T}, \tilde{t}_w$)
end function

[12] *Lucene* bietet eine Reihe von Möglichkeiten das *Scoring* durch diverse Parameter zu beeinflussen. Die vollständige Berechnung kann unter `http://lucene.apache.org/core/old_versioned_docs/versions/3_0_0/api/core/org/apache/lucene/search/Similarity.html` nachgeschlagen werden.

4.5.2 Berechnung der Term-Gewichte

Algorithmus 4.4 skizziert den schematischen Ablauf für die Berechnung der Term-Gewichte. In UniVideo werden dazu zunächst, für die Gewichtung der Tag-Empfehlungen, die k-ähnlichsten Videos eines Videos r mit Hilfe von Lucene beschafft. Hierfür wird der Titel von r in einzelne Terme zerlegt und die Stoppwörter herausgefiltert. Aus diesen Termen wird eine Suchabfrage (*Zend_Search_Lucene_Search_Query*) erstellt. Diese Abfrage wird an *Zend_Search_Lucene* gereicht und die Suche gestartet. Lucene liefert eine nach dem *Score* absteigend sortierte Liste mit den Ergebnis-Dokumenten zurück. Die Anzahl der Ergebnisse ist in diesem Setting auf maximal zehn Dokumente begrenzt ($k = 10$). Im Algorithmus ist dies etwas verkürzt dargestellt: N_r^k definieren wir als die Menge der Ergebnis-Dokumente, die uns von Lucene zurückgeliefert wird (Zeile 3). Über diese Ergebnisse wird darauffolgend in einer Schleife iteriert (Zeile 5). Jedes Ergebnis-Dokument enthält die *ID* seines zugehörigen Videos, mit der nun im Schleifenrumpf das Video-Objekt aus der Datenbank geladen werden kann. Im Algorithmus wird davon ausgegangen, dass *Lucene* das komplette Video-Objekt liefert, so dass dieser Schritt dort nicht dargestellt wird. In einer weiteren (inneren) Schleife (Zeile 7) wird nun über die Tags des gefundenen ähnlichen Videos iteriert, um deren Termgewichte zu berechnen. Die Berechnung der Term-Gewichte geschieht – wie bereits die k-Nachbarschaft durch Zend_Search_Lucene – mit dem *tf-idf*-Maß. Die Termhäufigkeit (*tf*) wird berechnet, indem das aktuell betrachtete Tag genommen wird und dessen Häufigkeit im Titel und der Beschreibung des aktuellen Videos gezählt wird (Zeile 8). Die invertierte Dokumentenhäufigkeit (*idf*) wird mit Hilfe der Lucene-Funktion `docFreq()` bestimmt (Zeile 9). Diese Funktion zählt die Dokumente im Dokumenten-Korpus, in denen ein bestimmter Term vorkommt. Das Gewicht des Terms ist das Produkt des *Scorings* des Videos mit der Termhäufigkeit *tf* und der invertierten Dokumentenhäufigkeit *idf* (Zeile 10). Die Terme werden in einer Liste zunächst gesammelt (Zeile 14), bis die Tags aller ähnlichen Videos gewichtet sind. Diese werden dann zwischen 0 und 1 normalisiert, indem jedes Gewicht durch das Maximum der Gewichte geteilt wird (Zeile 17 bis 19). Die ersten n Elemente in der Liste sind wieder die gewünschten Tag-Empfehlungen.

4.6 Hybrider Tag-Recommender durch gewichtetes Verschmelzen

Das Ziel des hybriden Tag-Recommenders ist es, die Ergebnisse der vier hier vorgestellten Tag-Recommender zu einem zusammenzufassen (siehe Abb. 4.1).

Eine einfache Möglichkeit einen hybriden Recommender umzusetzen, ist das gewichtete Verschmelzen (*weighted merging*) von verschiedenen Empfehlungs-Quellen als Linearkombination [Bur02]. Bei n Recommendern gibt es n verschiedene Mengen von

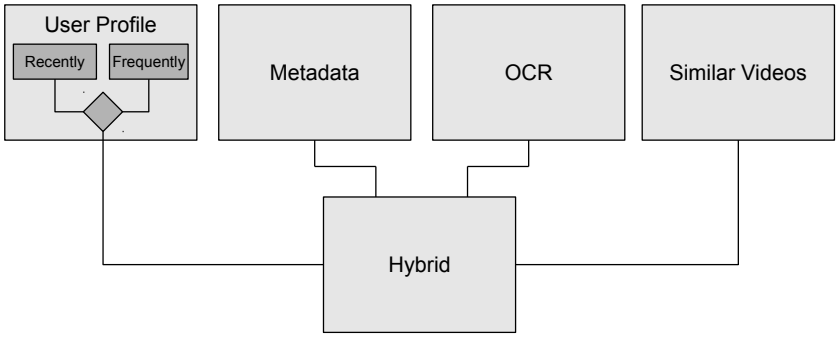


Abb. 4.1: Skizze des Hybriden Tag-Recommenders

Tag-Empfehlungen $\widetilde{T}_1(u,r),\ldots,\widetilde{T}_n(u,r)$ und n Gewichtungsfunktionen $f_1,\ldots,f_n$ für die Gewichtung der Terme der jeweiligen Empfehlungsmenge. Die Gesamtmenge als Vereinigung der einzelnen Empfehlungsmengen der unterschiedlichen Recommender $\widetilde{T}(u,r) := \bigcup_{i=1}^{n} \widetilde{T}_i(u,r)$ ist dann die Menge der Empfehlungen des hybriden Recommenders mit der Gewichtungsfunktion $f(t) := \sum_{i=1}^{n} \lambda_i \cdot f_i(t)$ mit $f_i(t) := 0$ wenn $t \notin \widetilde{T}_i(u,r)$ und $\sum_{i=1}^{n} \lambda_i = 1$ [JEHS09].

Wird davon ausgegangen, dass die n Recommender nach ihrer Ausführung Termvektoren $\vec{v}_1,\ldots,\vec{v}_n$ mit Termgewichten liefern, lässt sich diese Methode auch als eine Linearkombination der verschiedenen Ergebnisvektoren darstellen:

$$\vec{v} := \sum_{i=1}^{n} \lambda_i \cdot \vec{v}_i. \tag{4.5}$$

Auch hier gilt wieder $\sum_{i=1}^{n} \lambda_i = 1$. Die Skalare $\lambda_1,\ldots,\lambda_n$ sind nützlich als Stellschrauben, mit denen festgelegt wird, welcher Recommender wieviel Einfluss auf den hybriden Tag-Recommender hat. Dass die Summe aller Skalare 1 sein muss, sorgt dafür, dass die normalisierten Gewichte der Tag-Empfehlungen auch im Ergebnisvektor des hybriden Recommenders normalisiert sind.

4.6.1 Umsetzung in UniVideo

Bei der Umsetzung in UniVideo wird auf eine bestehende Datenstruktur von PHP zurückgegriffen. PHP bietet ein assoziatives Array als Datenstruktur an, das mit einer Hash-Map vergleichbar ist. Die Adressen der Einträge (Schlüssel) bestehen aus Zeichenketten. Da für die Recommender jedes Tag nur einmal vorkommen darf, bietet sich die Nutzung eines assoziativen Arrays an. Der Term wird als Schlüssel – also zur Adressierung des Feldes – genutzt und der Wert des Feldes ist das Gewicht des Terms.

In Listing 4.1 wird die Umsetzung des hybriden Recommenders mit PHP für UniVideo gezeigt. Konkret wird über die vier Algorithmus-Objekte iteriert (Zeile 6). Mit `setMax(-1)` wird dem Algorithmus mitgeteilt, dass es keine Begrenzung in der Anzahl der zurückgegebenen Empfehlungen geben soll. Die Methode `setScalar()` überreicht das zugehörige Skalar und die Methode `run()` führt den Algorithmus schließlich aus. Die Term-Gewichte werden direkt in den einzelnen Algorithmen mit dem Skalar multipliziert, so dass die Ergebnisse bereits als Skalarprodukt $\lambda_i \cdot \vec{v}_i$ zurückgegeben werden. Da die Ergebnisse der einzelnen Recommender Listen mit Objekten vom Typ *RecommendedTerm* sind, muss, bevor diese addiert werden können, aus dieser *ArrayList* ein assoziatives Array (Term-Map) erstellt werden (Zeile 10), mit der die Operationen leichter auszuführen sind. Ist dies geschehen, kann die Term-Map mit der Term-Map des vorherigen Iterationsschrittes addiert werden (Zeile 11).

In der Methode `termMapAddition($a, $b)` wird die Addition der Vektoren `$a` und `$b` ausgeführt (ab Zeile 23). Hier wird mit einer *foreach*-Schleife über alle Einträge in `$a` iteriert. Für jeden Iterationsschritt wird der Schlüssel aus `$a` als Variable `$term` zur Verfügung gestellt und das Gewicht als Variable `$weight`. Es wird geprüft, ob der Schlüssel `$term` aus `$a` in `$b` schon vorhanden ist. Ist das der Fall, wird das Gewicht von `$a` und `$b` addiert und in `$b[$term]` gespeichert. Ist der Term in `$b` nicht enthalten, wird ein neues Feld mit dem Schlüssel `$term` und den Wert `$weight` in `$b` hinzugefügt. Nachdem die Schleife durchgelaufen ist, wird `$b` zurückgegeben.

Sind alle Term-Maps miteinander addiert, muss die resultierende Term-Map in eine *ArrayList* mit *RecommendedTerm*-Objekten umgewandelt werden. Die nun wieder erhaltene *ArrayList* mit Term-Objekten wird dann nach ihren Gewichten sortiert vom Algorithmus verkürzt auf die Länge n als (endgültiges) Ergebnis zurückgegeben.

Listing 4.1: Umsetzung des hybriden Recommenders

```
1  class HYBRID extends AbstractAlgo {
2    ...
3    public function load_recommended_terms_for_object() {
4      ...
5      $i = 0; $res = array();
6      foreach($algorithm as $algo) {
7        $algo->setMax(-1); //Keine Begrenzung
8        $algo->setScalar($scalar[$i++]);
9        $algo->run();
10       $v = $this->arrayListToTermMap($algo->get_recommended_terms());
11       $res = $this->termMapAddition($res, $v);
12     }
13     ...
14   }
15   protected function arrayListToTermMap(ArrayList $list) {
16     $a = array();
17     for($list->rewind(); $list->valid(); $list->next()) {
```

```
18        $t = $list->current();
19        $a[$t->toString()] = $t->getScore();
20      }
21      return $a;
22    }
23    protected function termMapAddition($a, $b) {
24      foreach($a as $term => $weight) {
25        if( ! array_key_exists($term, $b) )
26          $b[$term] = $weight; //Neues Feld
27        else
28          $b[$term] = $b[$term] + $weight; //Addition
29      }
30      return $b;
31    }
32    ...
33  }
```

4.7 Zusammenfassung

Es wurden vier Tag-Recommender für UniVideo vorgestellt, die verschiendene Quellen zur Generierung von Tag-Empfehlungen heranziehen. Dabei nutzt der Recommender *User Profile*, die in den vergangenen 24 Stunden genutzten Tags eines Benutzers (*recently used tags*) und wenn es diese nicht gibt, die insgesamt am häufigsten verwendeten Tags dieses Nutzers (*frequently used tags*). Der inhaltsbasierte Tag-Recommender *Metadata* nutzt die Metadaten des Videos als Quelle und der inhaltsbasierte Tag-Recommender *OCR* im Video gefundene Textpassagen. Der Algorithmus *Similar Videos* lokalisiert ähnliche Videos und schlägt die Tags dieser als Empfehlungen vor. Der hybride Tag-Recommender ist eine Linearkombination der Term-Vektoren der Teil-Recommender, wobei jeder Term-Vektor gleich stark gewichtet ist.

5 Evaluation und Auswertung

In diesem Kapitel werden die fünf vorgestellten Algorithmen bewertet. Die Methodik und die Datenbasis wird in Abschnitt 5.1 diskutiert. In Abschnitt 5.2 folgt die Bewertung nach der Offline- und in Abschnitt 5.3 die Bewertung nach der Online-Evaluation.

5.1 Methoden und Datenbasis

5.1.1 Datenbasis

Die Evaluation teilt sich in eine Offline- und in eine Online-Evaluation. Für die Offline-Evaluation wurden die Daten verwendet, die von den Benutzern erzeugt wurden, bevor der Tag-Recommender eingesetzt worden ist. Zu dem Zeitpunkt als der Tag-Recommender in Betrieb genommen wurde, waren 642 hochgeladene Videos vorhanden. Davon wurden die 600 neueren als Testdaten verwendet. Besonders unter den ersten 100 Datensätzen sind teilweise Videos von ihren Eigentümern gar nicht verschlagwortet worden. Genau genommen wurden lediglich 530 von den insgesamt 600 ausgewählten Videos verschlagwortet. Dass einige Nutzer ihre Videos nicht verschlagworten, wurde schon früh als Problem erkannt. In UniVideo wurde deswegen eine Sperre eingebaut, die beim Hochladen von Videos die Eingabe von Tags erzwingt. Solange der Nutzer keine Tags eingetragen hat, durfte das Video nicht hochgeladen werden. Diese Sperre führte wiederum dazu, dass teilweise unpassende Tags eingegeben wurden, weil die Nutzer möglicherweise nicht genügend Zeit hatten, um sich passende Tags zu überlegen, oder weil der kognitive Aufwand zur manuellen Indexierung einfach zu hoch war. Ein weiteres Problem ist die teilweise zu geringe Indexierungsstufe, da oft lediglich pro Video nur ein Tag verwendet wurde. Außerdem hat es den Anschein, dass viele Nutzer den Sinn des Taggens nicht erkannt haben könnten, da nicht selten einfach der Titel kopiert und in das Eingabefeld für die Tags eingefügt wurde. Nicht wenige Tags bestehen deswegen aus Präpositionen, Artikeln und anderen Stoppwörtern, die ein guter Recommender auch nicht vorschlagen sollte. Diese Datenbasis bietet einerseits vielfältige Gründe für die Entwicklung eines Tag-Recommenders und andererseits ist sie allein nicht ausreichend geeignet, um die Qualität des Tag-Recommenders zu evaluieren.

Um also ein realistischeres Bild über die Qualität des Recommenders zu bekommen, wurde der Recommender zusätzlich im Betrieb getestet und die generierten Tag-Empfehlungen

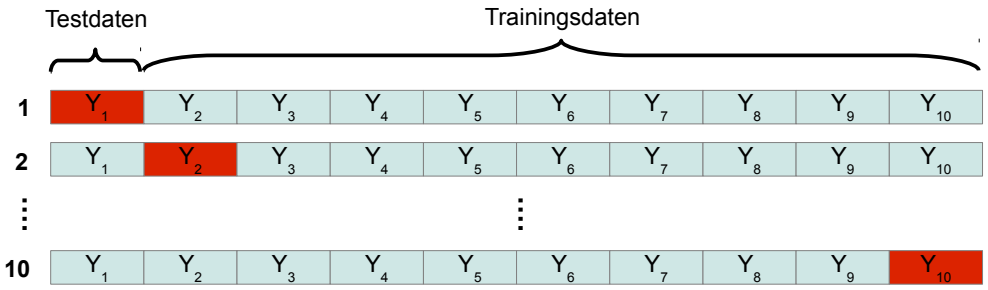


Abb. 5.1: 10-fache Kreuzvalidierung

gespeichert (vgl. Abschnitt 3.3.2). In einem Zeitraum von vier Wochen wurden 86 Videos hochgeladen, die mit unterschiedlichsten Tags von ihren Nutzern annotiert worden sind. Diese Daten dienen hier als Grundlage für die Online-Evaluation.

5.1.2 Methodik Offline-Evaluation

Überlicherweise werden im Information Retrieval die Daten zur Evaluation in zwei Teile geteilt: eine Menge Trainingsdaten, die für die Berechnung benötigt wird und eine kleinere Menge Testdaten, für die die Klassifikatoren (in diesem Fall die Tag-Recommender-Algorithmen) getestet wird. Da oftmals die Datenmenge der Test- und/oder Trainingsmenge nicht ausreicht, um eine verlässliche Schätzung zu erhalten, werden die Test- und (Teil-)Trainingsdaten sukzessive getauscht. Dazu wird die zur Verfügung stehende Datenmenge in $k \leq N$ möglichst gleichgroße Teilmengen $Y_1, \ldots, Y_k$ geteilt. In k Testläufen ist die Datenmenge Y_i immer die Testmenge und die $k-1$ übrigen Datenmengen $\{Y_1, \ldots, Y_k\} \setminus \{Y_i\}$ dienen als Trainingsdaten (vgl. Abb. 5.1).

Für die Evaluation der Offline-Datensätze wurde die 10-fache Kreuzvalidierung ($k = 10$) genutzt. Die Testdaten wurden hierfür in 10 gleichgroße Teile (*folds*) mit einer Anzahl von jeweils 60 Datensätzen geteilt. Die jeweils übrigen 540 Datensätze dienten als Trainingsdaten.

Da der Algorithmus *Similar Videos* zur Berechnung der Tag-Vorschläge *Lucene* nutzt (vgl. Abschnitt 4.5), musste – damit eine konsistente Schätzung möglich wurde – der Such-Index ebenfalls in 10 Teile geteilt werden. Da sich der Such-Index nicht ohne weiteres aufteilen lässt, wurden 10 Such-Indizes $I_1, \ldots, I_{10}$ angelegt, die jeweils aus den 540 Trainingsdatensätzen $\{Y_1, \ldots, Y_{10}\} \setminus \{Y_i\}$ des jeweiligen *folds* i bestehen, bei denen die Menge der Testdaten Y_i ausgespart bleibt.

Für die Bewertung der Algorithmen wurden als Maße Precision, Recall und das F-Maß verwendet (vgl. Abschnitt 2.4). Für jede Testdatenmenge eines *folds* wurden Precision und

Recall für 1 bis 10 Tag-Empfehlungen berechnet und in einer Variable aufsummiert. Nachdem alle Tag-Empfehlungen des *folds* berechnet worden sind, wurden diese Werte durch die Anzahl der Elemente geteilt. Auf diese Weise konnte für jede der 10 Testdatenmengen für jeweils 1 bis 10 Tag-Empfehlungen Precision und Recall als arithmetisches Mittel ermittelt werden. Nach Durchlauf aller *folds* wurden diese Werte nochmals gemittelt, um die Mittelwerte der gesamten Datenmenge zu erhalten. Aus den Mittelwerten von Precision und Recall wurde dann das F-Maß für die jeweilige Anzahl an Empfehlungen bestimmt.

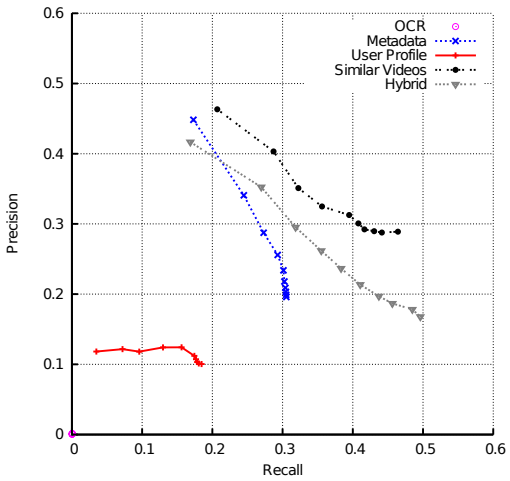
5.1.3 Methodik Online-Evaluation

Für die Online-Evaluation wurde keine Kreuzvalidierung angewandt, da bei der Online-Berechnung der Tag-Empfehlungen das jeweilige Video in der Trainingsmenge nicht vorhanden sein konnte. Diese online berechneten Tag-Empfehlungen wurden mit den genutzten Tags und den eingegebenen Video-Informationen direkt gespeichert (vgl. Abschnitt 3.3.2). Diese gespeicherten Tag-Empfehlungen können also direkt zur Bewertung herangezogen werden. Für alle 86 Videos wird das arithmetische Mittel für Precision, Recall und F-Maß bestimmt.

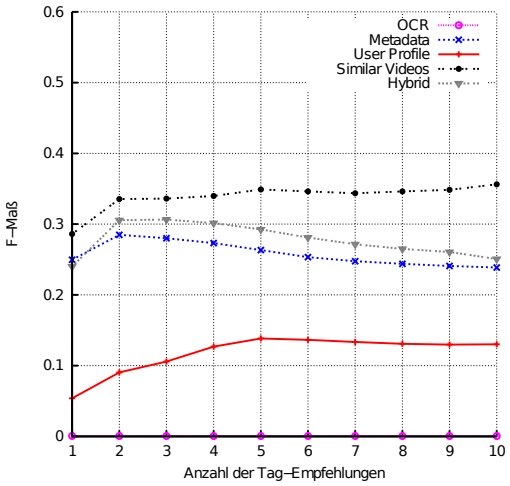
5.2 Offline-Evaluation

Tabelle 5.1 dokumentiert die Ergebnisse der Offline-Evaluation, die in Abb. 5.2 nochmal graphisch aufgearbeitet sind. Im Diagramm 5.2a wird an der X-Achse der Recall und an der Y-Achse die Precision gemessen. Für jeden Recommender werden diese Werte für ein bis zehn Tag-Vorschläge abgetragen, wobei mit steigender Anzahl der Tag-Empfehlungen auch der Recall steigt. Das Diagramm 5.2b zeigt das Verhalten der Algorithmen in Bezug auf das F-Maß. An der X-Achse wird hier die Anzahl der Empfehlungen abgetragen; an der Y-Achse wird das F-Maß gemessen. Dort nicht sichtbar ist die hohe Standardabweichung aller (Teil-)Recommender. Diese lässt sich auf die geringe Anzahl der zur Verfügung stehenden Testdaten zurückführen.

Die Algorithmen verhalten sich in der Offline-Evaluation sehr unterschiedlich. Die geringe Anzahl an Testdaten ermöglicht leider kein differenziertes Bild. Insgesamt lässt sich allerdings ein positives Resultat feststellen, da die Recommender teilweise erstaunlich hohe Genauigkeiten (Precision) und Trefferquoten (Recall) aufzeigen. In den folgenden Abschnitten werden die Ergebnisse der einzelnen Recommender diskutiert.



(a) Precision und Recall aller fünf Recommender in Abhängigkeit zur Anzahl der vorgeschlagenen Tags



(b) F-Maß aller fünf Recommender in Abhängigkeit zur Anzahl der vorgeschlagenen Tags

Abb. 5.2: Ergebnisse der Offline Evaluation bei 10-facher Kreuzvalidierung

5.2.1 User Profile

Mit einem Recall von knapp 14 Prozent und einer Precision von 6,5 Prozent bei 10 Tag-Empfehlungen, liefert der Algorithmus User Profile vergleichsweise schlechte Ergebnisse. Diese Ergebnisse lassen sich aber auf die Evaluations-Methode der Kreuzvalidierung zurückführen. Da *User Profile* die Berechnung auf Basis der Tags des jeweiligen Nutzers (*frequently used tags*), beziehungsweise auf Basis der Tags des jeweiligen Nutzers der vergangenen 24 Stunden (*recently used tags*) stützt, ist es sehr wahrscheinlich, dass zumindest ein Teil davon nicht in der Menge der Trainingsdaten, sondern in der Menge der Testdaten liegt und somit für die Berechnung nicht berücksichtig werden kann. Im Falle von *recently used tags* ist das sogar sehr wahrscheinlich, da dieser Algorithmus auf Daten zurückgreift, die in einem sehr kurzen Zeitraum liegen. Um eine realistischere Schätzung zu erhalten, wurde die Anzahl der *folds* bei der Kreuzvalidierung exemplarisch erhöht. Mit $k = 60$, also mit 60-facher Kreuzvalidierung, ergab dies bei 10 vorgeschlagenen Tags bereits einen Recall von 0.2676 (bei $k = 10$: 0.1847) und eine Precision von 0.1787 (bei $k = 10$: 0.1004). Bei $k = 100$ betrug der Recall sogar 0.3325 und die Precision 0.2335. Diese Zahlen lassen ein ganz anderen Schluss zu, als die der 10-fachen Kreuzvalidierung. Die schlechten Werte dieses Algorithmus lassen sich also (nur) in diesem speziellen Fall auf die 10-fache Kreuzvalidierung zurückführen. Die Bewertung von *User Profile* anhand der Zahlen der 10-fachen Kreuzvalidierung muss demnach mit einer gewissen Vorsicht erfolgen.

Tab. 5.1: Ergebnisse der Offline-Evaluation bei 10-facher Kreuzvalidierung

	User Profile			**Metadata**			**OCR**		
#Tags	Recall	Precision	F-Maß	Recall	Precision	F-Maß	Recall	Precision	F-Maß
1	0.0348	0.1183	0.0538	0.1730	0.4483	0.2497	0.0004	0.0017	0.0007
2	0.0721	0.1217	0.0905	0.2448	0.3408	0.2849	0.0004	0.0008	0.0006
3	0.0959	0.1181	0.1058	0.2730	0.2875	0.2801	0.0004	0.0008	0.0006
4	0.1297	0.1242	0.1269	0.2930	0.2558	0.2732	0.0004	0.0008	0.0006
5	0.1562	0.1243	0.1384	0.3012	0.2339	0.2633	0.0004	0.0008	0.0006
6	0.1744	0.1121	0.1365	0.3026	0.2180	0.2534	0.0004	0.0008	0.0006
7	0.1767	0.1073	0.1335	0.3040	0.2089	0.2476	0.0004	0.0008	0.0006
8	0.1783	0.1033	0.1308	0.3046	0.2033	0.2439	0.0004	0.0008	0.0006
9	0.1807	0.1012	0.1298	0.3051	0.1991	0.2409	0.0004	0.0008	0.0006
10	0.1847	0.1004	0.1301	0.3054	0.1959	0.2387	0.0004	0.0008	0.0006

(a) *User Profile*, *Metadata* und *OCR*

	Similar Videos			**Hybrid**		
#Tags	Recall	Precision	F-Maß	Recall	Precision	F-Maß
1	**0.2070**	**0.4633**	0.2861	0.1686	0.4167	0.2400
2	**0.2871**	**0.4033**	0.3354	0.2696	0.3525	0.3055
3	**0.3224**	**0.3511**	0.3361	0.3185	0.2953	0.3065
4	**0.3562**	**0.3249**	0.3398	0.3550	0.2619	0.3015
5	**0.3948**	**0.3127**	0.3490	0.3833	0.2366	0.2926
6	0.4080	**0.3007**	0.3463	**0.4105**	0.2136	0.2810
7	0.4166	**0.2923**	0.3436	**0.4371**	0.1970	0.2716
8	0.4304	**0.2897**	0.3463	**0.4563**	0.1867	0.2650
9	0.4414	**0.2878**	0.3484	**0.4848**	0.1782	0.2606
10	0.4643	**0.2890**	0.3563	**0.4962**	0.1678	0.2508

(b) *Similar Videos* und *Hybrid*

5.2.2 Metadata

Die Ergebnisse des Algorithmus *Metadata* bestätigen die These von Lipczak und Milios, dass der Titel und die Beschreibung eine gute Quelle für Tag-Empfehlungen sind [LM10]. Dass die Precision-Recall-Kurve bei steigender Anzahl von Tag-Empfehlungen so stark fällt, lässt sich durch die meistens kurzen Titel und Beschreibungen begründen. Die Ergebnisse sind dennoch sehr gut. Selbst bei einer Anzahl von 10 Empfehlungen, liegt die Precision noch bei knapp 20 Prozent. Da im Betrieb von UniVideo ein hybrider Recommender zum Einsatz kommt und dadurch zusätzlich andere Quellen herangezogen

werden, sind vermutlich die ersten Tag-Empfehlungen von hoher Relevanz.

5.2.3 OCR

Die Ergebnisse des *OCR*-Algorithmus liegen annähernd bei 0. Dies lässt auf drei wesentliche Probleme zurückführen:

1. Die *OCR*-Software kann nur in Videos die Text enthalten diesen auch herausfiltern. Von den 600 Testdatensätzen konnte lediglich in 120 Videos überhaupt Text mit *OCR* extrahiert werden.
2. Die Berechnung der Term-Gewichte wird bewerkstelligt, indem die Anzahl der Nutzung des jeweiligen Terms in den Metadaten und als Tag durch die Anzahl der Vorkommenshäufigkeit nur in den Metadaten geteilt wird (vgl Abschnitt 4.4). Tritt ein per OCR gefundener Term in den Metadaten und/oder den dazugehörigen Tags nicht auf, ist sein Gewicht demnach 0 und wird in der Liste der Tag-Empfehlungen hinten einsortiert. Die Wahrscheinlichkeit, dass dieser Term als Tag vorgeschlagen wird, ist also sehr gering.
3. Die Ausbeute der mit der OCR-Software extrahierten Terme ist zu gering.

Das erste Problem ist nicht lösbar und muss dementsprechend hingenommen werden. Wo kein Text ist, kann auch keiner extrahiert werden.

Das zweite Problem könnte mit einem anderen Term-Gewichtungsverfahren gelöst werden. Mit dem *tf-idf*-Maß könnte der Algorithmus bessere Ergebnisse liefern. Dabei ließe sich *tf-idf* auf zwei unterschiedliche Weisen einsetzen. Eine Möglichkeit wäre *tf-idf* auf die durch *OCR* extrahierten Terme anzuwendet und eine andere Möglichkeit wäre die Anwendung von *tf-idf* auf die Metadaten.

Die Ausbeute der extrahierten Terme ließe sich erhöhen, indem eine bessere Software eingesetzt wird. Viele *OCR*-Programme können durch Trainieren verbessert werden. Ein weiterer wesentlicher Grund für die schlechte Ausbeute könnte ein zu geringer Kontrast zwischen Text und Hintergrund sein. In [YSL+11] wurde ein Verfahren beschrieben, welches eine dynamische Kontrast-und Helligkeitsanpassung ermöglicht. Nach der Anwendung dieses Algorithmus soll der Text schwarz auf weißen Hintergrund erscheinen, was die Ausbeute der OCR-Software verbessern dürfte.

5.2.4 Similar Videos

UniVideo ist eine enge Folksonomy (vgl. Abschnitt 2.1), da Nutzer keine gemeinsamen Ressourcen mit Tags annotieren, sondern immer nur ihre eigenen. Ein Tag-Recommender auf Basis des Kollaborativen Filterns wäre deswegen möglicherweise keine optimale

Wahl gewesen, da sich dieser nur mit Hilfe des gemeinsam genutzten Tag-Vokabulars herstellen ließe (vgl. Abschnitt 2.3.3). Die Tags der k-ähnlichsten Ressourcen statt die der k-ähnlichsten Nutzer zu verwenden hat sich als gute Strategie herausgestellt, wie die Ergebnisse belegen. Der Algorithmus *Similar Videos* liefert sowohl bei der Precision also auch beim Recall die besten Ergebnisse der nicht-hybriden Recommender.

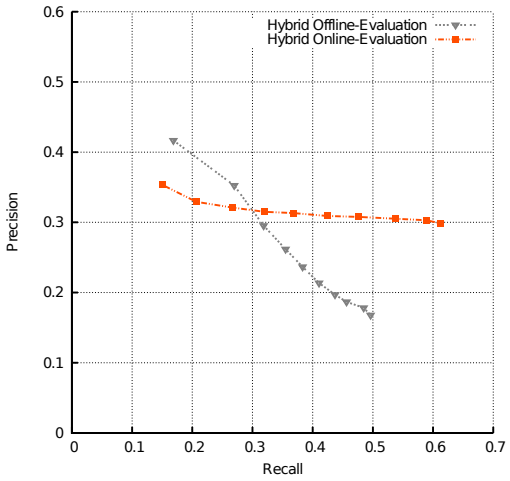
5.2.5 Hybrid

Der hybride Algorithmus liefert, bei gleich starker Gewichtung aller Algorithmen ($\lambda_i = 0.25$), ebenfalls gute Ergebnisse, fällt aber bei der Precision hinter *Similar Videos* deutlich zurück. Im Recall sind beide Algorithmen etwa gleich stark bei bis zu sieben Tag-Empfehlungen. Bei mehr als sieben Tag-Empfehlungen wird dann der hybride Recommender etwas besser. Wünschenswert wäre, dass der hybride Algorithmus immer bessere Ergebnisse liefert als die einzelnen – zumindest beim F-Maß. Erklären ließe sich das Verhalten des hybriden Recommenders dadurch, dass die einzelnen Tag-Recommender viele gemeinsame Tags finden und diese jeweils hoch gewichtet werden. Die hohe Schnittmenge könnte dafür sorgen, dass besonders viele niedrig gewichtete Terme als Empfehlungen vorgeschlagen werden, die zum starken Abfall der Precision führen. Die durch die 10-fache Kreuzvalidierung verursachte Verminderung der Werte des *User Profile*-Recommenders, ist vermutlich ebenso für diese Werte verantwortlich.

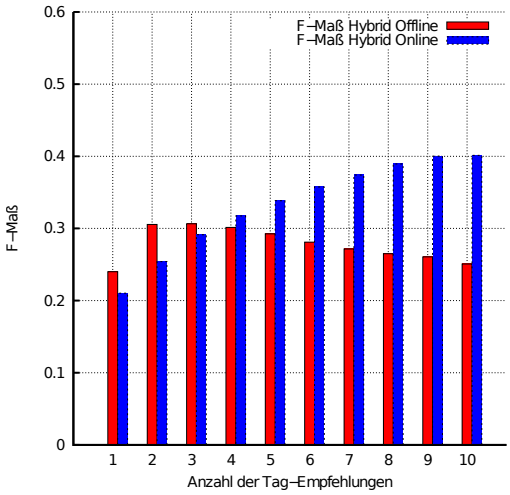
5.3 Online-Evaluation

Tab. 5.2: Ergebnisse der Online- und Offline Evaluation des hybriden Tag-Recommenders im Vergleich

	Offline Evaluation			**Online Evaluation**		
#Tags	Recall	Precision	F-Maß	Recall	Precision	F-Maß
1	0.1686	0.4167	**0.2400**	0.1497	0.3537	0.2103
2	0.2696	0.3525	**0.3055**	0.2067	0.3293	0.2540
3	0.3185	0.2953	**0.3065**	0.2667	0.3210	0.2913
4	0.3550	0.2619	0.3015	0.3199	0.3150	**0.3174**
5	0.3833	0.2366	0.2926	0.3681	0.3132	**0.3385**
6	0.4105	0.2136	0.2810	0.4239	0.3091	**0.3575**
7	0.4371	0.1970	0.2716	0.4767	0.3077	**0.3740**
8	0.4563	0.1867	0.2650	0.5381	0.3051	**0.3894**
9	0.4848	0.1782	0.2606	0.5896	0.3029	**0.4002**
10	0.4962	0.1678	0.2508	0.6127	0.2984	**0.4013**



(a) Precision und Recall der Online- und Offline-Evaluation des hybriden Recommenders in Abhängigkeit zur Anzahl der vorgeschlagenen Tags



(b) F-Maße der Online- und Offline-Evaluation des hybriden Recommenders in Abhängigkeit zur Anzahl der vorgeschlagenen Tags

Abb. 5.3: Ergebnisse der Online- und Offline-Evaluation im Vergleich

Wie die Offline-Evaluation den Einfluss eines Tag-Recommenders auf das menschliche Verhalten beim Taggen nicht berücksichtigen kann, so kann die Online-Evaluation kein Bild frei von Suggestion liefern. Der Grad der Beeinflussung des Tag-Recommenders auf den Menschen lässt sich nicht messen, weswegen der Vergleich der herkömmlichen Werte von Offline- und Online-Evaluation notwendig ist, um ein realistisches Gesamtbild von der Qualität der generierten Tag-Empfehlungen zu erhalten. Dies bestätigt sich beim Vergleich der Ergebnisse von Online- und Offline-Evaluation in Tabelle 5.2 und Abbildung 5.3.

Auch für die Online-Evaluation sind alle Teil-Recommender gleich stark gewichtet. Da die Herkunft der vom Nutzer übernommenen Tag-Empfehlungen nicht gespeichert wurde, lässt sich jedoch nicht nachvollziehen, welche Teil-Recommender für den Nutzer die relevantesten Empfehlungen liefern. Dennoch lassen die Ergebnisse einige Schlussfolgerungen zu: Am auffälligsten an den Ergebnissen ist das nur leichte Gefälle der Precision bei steigender Anzahl von Tag-Empfehlungen. Das deutet zum einen darauf hin, dass geringer gewichtete Terme für den Nutzer fast genauso oft passable Ergebnisse liefern, wie die im Term-Vektor weiter vorn liegenden höher gewichteten Tags. Zum anderen korrigiert das die vergleichsweise stark fallende Precision-Recall-Kurve der Offline-Evaluation, wie in Abbildung 5.2a veranschaulicht wird. Das F-Maß steigt bei steigender Anzahl von Tag-Empfehlungen stetig. Wird dieses Maß als Grundlage genommen, wird der hybride

Recommender bei steigender Anzahl von Tag-Empfehlungen sogar besser, während es bei der Offline-Evaluation beim hybriden und allen Teil-Recommendern (mit Ausnahme des Algorithmus *Similar Videos*) anfangs kurz steigt und dann stetig fällt.

5.4 Zusammenfassung

Die Online- und Offline-Evaluation hat für alle (Teil-)Recommender – mit Ausnahme von *OCR* – passable Ergebnisse geliefert. Um abschließend eine realistische Bewertung vornehmen zu können, muss die Online-Evaluation über einen längeren Zeitraum mit einer höheren Anzahl an Datensätzen durchgeführt werden und mit den Ergebnissen der Offline-Evaluation nochmals verglichen werden.

6 Fazit und Ausblick

Gegenstand dieser Arbeit war die Entwicklung eines Tag-Recommenders für Video-Ressourcen im Video-Portal UniVideo, der mehrere Quellen zur Generierung von Tags nutzt. Vier verschiedene Ansätze wurden untersucht:

- Empfehlungen auf Basis zuletzt genutzter Tags und häufig genutzter Tags des Nutzers (*User Profile*),
- inhaltsbasierte Empfehlungen mit Titel und Beschreibung als Quelle (*Metadata*),
- inhaltsbasierte Empfehlungen mit Texterkennung aus Standbildern des Videos als Quelle (*OCR*) und
- Tag-Empfehlungen aus ähnlichen Videos, die mit Hilfe der Text-Suchmaschine *Lucene* ermittelt werden (*Similar Videos*).

Im Folgenden werden einige Änderungen bzw. Anpassungen der Recommender aufgezählt, die im Rahmen dieser Arbeit nicht mehr untersucht werden konnten. Nach derzeitigem Kenntnisstand könnten die Änderungsvorschläge möglicherweise noch Verbesserungen herbeiführen und für zukünftige Untersuchungen entsprechend interessant sein.

User Profile. Im Algorithmus *User Profile* wird zunächst geprüft, ob der Nutzer in den vergangenen 24 Stunden bereits Videos hochgeladen und verschlagwortet hat (*recently used tags*). Ist dies der Fall, bekommt er diese vorgeschlagen. Ist dies nicht der Fall, werden dem Nutzer seine am häufigsten genutzten Tags vorgeschlagen. Diese harte Trennung geht bedingungslos davon aus, dass der Nutzer, wenn er innerhalb von 24 Stunden mehrere Videos hochlädt, diese immer das gleiche oder ähnliche Thema haben und deswegen die gleichen Tags relevant sind. Statt einer Entweder-Oder-Verzweigung würde sich hier eine Verschmelzung anbieten, so dass dem Nutzer immer die Tags der vergangenen 24 Stunden (sofern vorhanden) *und* seine populärsten Tags vorgeschlagen werden.

Metadata. Derzeit spiegeln die berechneten Termgewichte dieses Recommenders das Verhältnis aus Vorkommenshäufigkeit eines Terms in den Metadaten (Titel oder Beschreibung) und dessen Vorkommenshäufigkeit in den Metadaten *und* den Tags wider. Auf diese Weise werden Terme aus den Metadaten vorgeschlagen, die (am häufigsten) als Tags genutzt worden sind. Die Nutzung des *tf-idf*-Maßes könnte zu einer Verbesserung führen, da

auf diese Weise noch nicht als Tags verwendete Terme vorgeschlagen werden. Würde jedes Wort aus dem Titel und der Beschreibung eines Videos mit *tf-idf* gewichtet werden, dann würden weniger häufig vorkommende Terme ein höheres Gewicht bekommen, als solche die häufiger genutzt wurden. Dies könnte dazu führen, dass beispielsweise Abkürzungen eines Sachverhalts als Tags vorschlagen werden, die bis dahin nicht im Tag-Vokabular enthalten waren. Auch hier könnte eine Verschmelzung beider Herangehensweisen sinnvoll sein.

OCR. Für den *OCR*-Recommender wurde als Verbesserung bereits ein Algorithmus in Abschnitt 5.2.3 vorgeschlagen, der den Kontrast zwischen Hintergrund und Schrift erhöht und dadurch eine höhere Ausbeute bei der Extrahierung von Text aus den Standbildern des Videos verspricht [YSL+11]. Allerdings hat der *OCR*-Recommender bereits das Problem, dass die Gewinnung der Tags zeitkritisch ist, weswegen er in zwei zeitlich voneinander getrennten Schritten ausgeführt wird (vgl. Abschnitt 4.4). Die Zerlegung in 20 Standbilder und die Anwendung der OCR-Software führt bereits jetzt zu einer Verzögerung im Upload-Prozess. Je nach Laufzeit des *Algorithmus zur Erhöhung des Kontrastes* könnte sich dieses Problem weiter zuspitzen. Um dies zu umgehen, wäre zu überprüfen, ob aus den einzelnen *Chunks*, in die die Videodatei beim Upload zerlegt wird, bereits Standbilder extrahiert werden können. Auf diese Weise könnte die OCR-Software während des – in der Regel ohnehin langwierigen – Uploads die ersten Standbilder bereits nach Textpassagen durchsuchen. Dieses Vorgehen dürfte den gesamten Prozess verkürzen.

Hybrid. Der hybride Tag-Recommender ist eine Linearkombination aus den Termvektoren der Teil-Recommender. Dabei wird der Termvektor $\vec{v}_i$ jedes Teil-Recommenders mit einem zugeordneten Skalar λ_i multipliziert (wobei $\sum_{i=1}^{n} \lambda_i = 1$), das als Gewicht des Teil-Recommenders verstanden werden kann. Momentan sind alle $\lambda_i = 0.25$, also gleich stark gewichtet. Um bessere Ergebnisse zu erhalten, kann die optimale Verteilung der Gewichte bestimmt werden, sodass die Summe dieser immer 1 bleibt. Dafür müssten z.B. alle F-Maße bei zehn Tag-Vorschlägen bei unterschiedlicher Verteilung der λ_i bestimmt werden. Die Kombination der λ_i, die das maximale F-Maß hervorbringt, wäre die optimale Gewichtung der Term-Vektoren der einzelnen Teil-Recommender.

Ein Hauptmotiv für die Entwicklung eines Tag-Recommenders war die geringe Indexierungstiefe. Viele Nutzer haben oft nur ein Tag oder gar kein Tag an ihre Ressource geschrieben. Seit der Tag-Recommender implementiert ist, werden durchschnittlich pro Video 3.9884 Tags eingeben. Zuvor lag dieser Wert bei 2.9359. Hier zeichnet sich die Tendenz ab, dass der Recommender den Nutzer dazu animiert mehrere Tags zu verwenden. Ob die genutzten Tags nun tatsächlich besser sind, hängt von der subjektiven Wahrnehmung ab und lässt sich nicht ohne weiteres messen.

Abschließend kann festgehalten werden, dass alle fünf Recommender praxistauglich umgesetzt werden konnten. Die Evaluation mit Testdaten (Offline-Evaluation) und die Bewertung über einen Zeitraum von vier Wochen im Live-Betrieb (Online-Evaluation) bestätigt diese Annahme. Insbesondere die Online-Evaluation zeigt tendenziell eine hohe Akzeptanz der Nutzer für die vorgeschlagenen Tags. Das zeigt sich daran, dass bei hoher Anzahl von Tag-Vorschlägen im Mittel eine vergleichsweise hohe Precision gemessen werden konnte. Um abschließend eine endgültige Bewertung vornehmen zu können, muss zu einem späteren Zeitpunkt die Online-Evaluation wiederholt werden, wenn mehrere Datensätze als Bewertungsgrundlage vorliegen. Um die einzelnen Teil-Recommender besser bewerten zu können, ist es außerdem notwendig die Herkunft der gewählten Empfehlungen zu speichern. Auf diese Weise kann dann auch ermittelt werden, welche Teil-Recommender die für die Nutzer relevantesten Empfehlungen liefern.

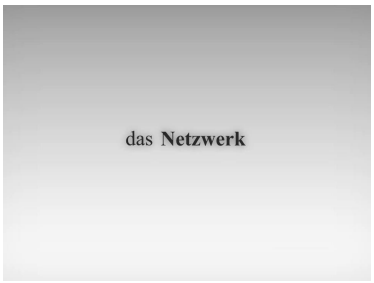
Für den Autor dieser Arbeit sind die Ergebnisse sehr zufriedenstellend und übertreffen seine vorherigen Erwartungen. Der hybride Tag-Recommender wird weiterhin im Einsatz bleiben. Sofern eine zukünftige Evaluation die Notwendigkeit zur Anpassung erfordert, werden die angesprochenen Verbesserungsvorschläge untersucht und ggf. umgesetzt.

A Anhang

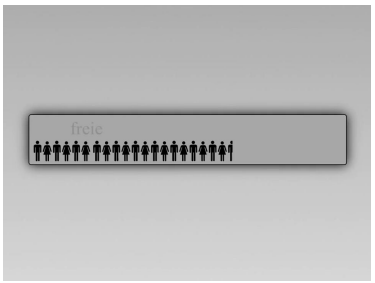
A.1 Mit OCR extrahierte Terme aus Video-Standbildern

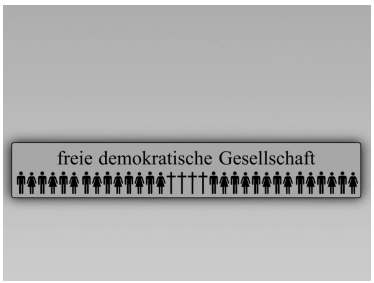
Die Abbildung A.1 zeigt die ersten 12 Standbilder, die aus dem Video „das Netzwerk“[1] erzeugt wurden. In der sich unterhalb des jeweiligen Bildes befindlichen Legende stehen die aus dem jeweiligen Standbild mit der Software *GOCR* gefundenen Terme (nach Filterung durch Rechtschreibkorrektur).

[1] Das vollständige Video kann unter `http://univideo.uni-kassel.de/video/993/` angeschaut werden.

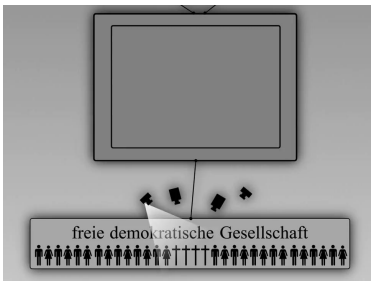


(a) das, Netzwerk

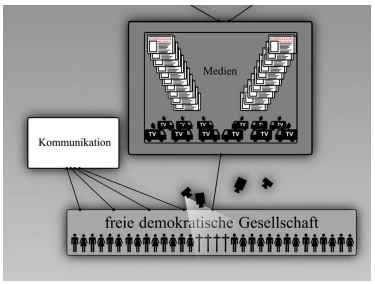




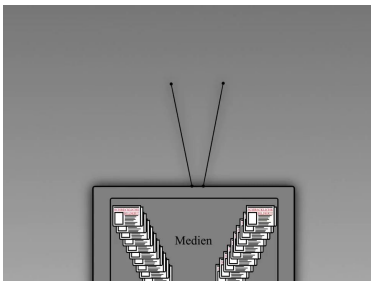
(c) freie, demokratische, Gesellschaft



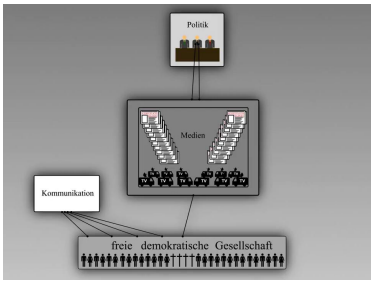
(d) freie, demo, Gesellschaft



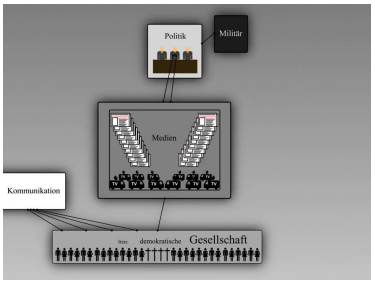
(e) freie

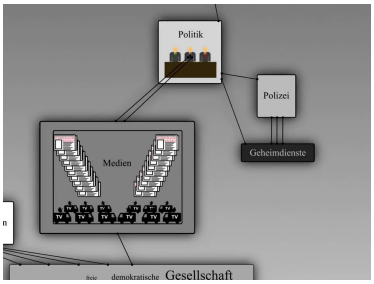


(f) Medien

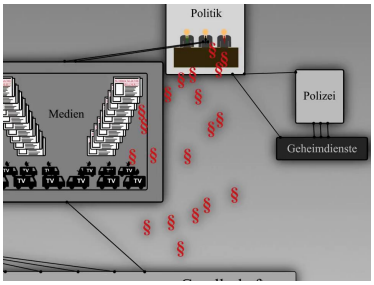


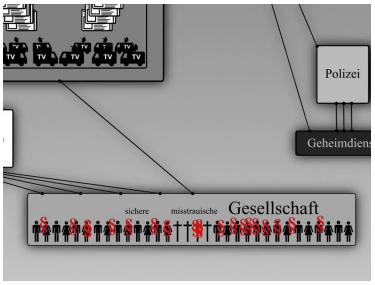
(g) freie, demokratische



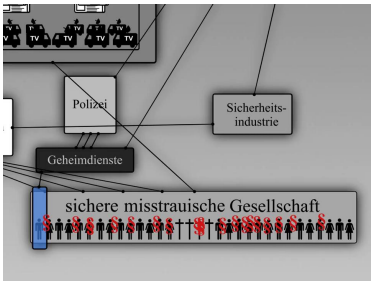


(i) Gesellschaft





(k) Polizei



(l) Gesellschaft

Abb. A.1: Gefundene Terme des Videos „das Netzwerk“

Literaturverzeichnis

[AN07] Ames, Morgan ; Naaman, Mor: Why We Tag: Motivations for Annotation in Mobile and Online Media. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 2007 (CHI '07), S. 971–980

[BBHS08] Blank, Michael ; Bopp, Thomas ; Hampel, Thorsten ; Schulte, Jonas: Social Tagging = Soziale Suche? In: Gaiser, Birgit (Hrsg.) ; Hampel, Thorsten (Hrsg.) ; Panke, Stefanie (Hrsg.): *Good Tags - Bad Tags. Social Tagging in der Wissensorganisation*. Münster, New York, München, Berlin : Waxmann, 2008, S. 85–97

[Bur02] Burke, Robin: Hybrid Recommender Systems: Survey and Experiments. In: *User Modeling and User-Adapted Interaction* 12 (2002), November, S. 331–370

[BYRN99] Baeza-Yates, Ricardo A. ; Ribeiro-Neto, Berthier: *Modern Information Retrieval*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1999

[CLP06] Cattuto, Ciro ; Loreto, Vittorio ; Pietronero, Luciano: *Collaborative Tagging and Semiotic Dynamics*. 2006

[JEHS09] Jäschke, Robert ; Eisterlehner, Folke ; Hotho, Andreas ; Stumme, Gerd: Testing and Evaluating Tag Recommenders in a Live System. In: *RecSys '09: Proceedings of the third ACM Conference on Recommender Systems*. New York, NY, USA : ACM, 2009, S. 369–372

[JMH+07] Jäschke, Robert ; Marinho, Leandro ; Hotho, Andreas ; Schmidt-Thieme, Lars ; Stumme, Gerd: Tag Recommendations in Folksonomies. In: Kok, Joost (Hrsg.) ; Koronacki, Jacek (Hrsg.) ; Mantaras, Ramon Lopez d. (Hrsg.) ; Matwin, Stan (Hrsg.) ; Mladenic, Dunja (Hrsg.) ; Skowron, Andrzej (Hrsg.): *Knowledge Discovery in Databases: PKDD 2007* Bd. 4702. Berlin / Heidelberg : Springer, 2007, S. 506–514

[Jäs11] Jäschke, Robert: *Dissertationen zur Künstlichen Intelligenz*. Bd. 332: *Formal Concept Analysis and Tag Recommendations in Collaborative Tagging Systems*. Heidelberg, Germany : Akademische Verlagsgesellschaft AKA, 2011

[LM10] Lipczak, Marek ; Milios, Evangelos: The impact of resource title on tags in collaborative tagging systems. In: *HT '10: Proceedings of the 21st ACM conference on Hypertext and hypermedia*. New York, NY, USA : ACM, 2010, S. 179–188

[MP07] Müller-Prove, Matthias: Taxonomien und Folksonomien - Tagging als neues HCI-Element (Taxonomies and Folksonomies - Tagging as a New HCI Element). In: *i-com* 6 (2007), Nr. 1, S. 14–18

[MP08] Müller-Prove, Matthias: Modell und Anwendungsperspektive des Social Tagging. In: Gaiser, Birgit (Hrsg.) ; Hampel, Thorsten (Hrsg.) ; Panke, Stefanie (Hrsg.): *Good Tags - Bad Tags*. Waxmann, 2008

[PG08] Panke, Stefanie ; Gaiser, Birgit: "With my head up in the clouds- Social Tagging aus Nutzersicht. In: Gaiser, Birgit (Hrsg.) ; Hampel, Thorsten (Hrsg.) ; Panke, Stefanie (Hrsg.): *Good Tags - Bad Tags* Bd. 47. Waxmann, 2008, S. 23–35

[SM83] Salton, G. ; McGill, M.: Introduction to Modern Information Retrieval. McGraw-Hill, 1983

[TSMST08] Tso-Sutter, Karen H. L. ; Marinho, Leandro B. ; Schmidt-Thieme, Lars: Tag-aware recommender systems by fusion of collaborative filtering algorithms. In: *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*. New York, NY, USA : ACM, 2008, S. 1995–1999

[Wal05] Wal, Thomas V.: *Explaining and Showing Broad and Narrow Folksonomies*. Blog post. `http://vanderwal.net/random/entrysel.php?blog=1635`. Version: February 2005

[Wal07] Wal, Thomas V.: *Folksonomy Coinage and Definition*. `http://vanderwal.net/folksonomy.html`. Version: 2007

[YSL+11] Yang, Haojin ; Siebert, Maria ; Lühne, Peter ; Sack, Harald ; Meinel, Christoph: Lecture Video Indexing and Analysis Using Video OCR Technology. In: *7th Int. Conf. on Signal Image Technology and Internet Based Systems (SITIS 2011), Track Internet Based Computing and Systems*, 2011