Universität Bielefeld

Projektbericht

Kommunikationsmedium E-Mail-Newsletter

XML-Analyse des monatlichen Newsletters einer Umwelt- und Menschenrechtsorganisation

Max Harder
Matr.-Nr.: 2919411
max.harder@uni-bielefeld.de

Politikwissenschaft / Bachelor of Arts: Kernfach (fw)
Texttechnologie und Computerlinguistik / Bachelor: Nebenfach (fw)

Modul 23-TXT-BaCL4 Informationsstrukturierung
230010 Informationsstrukturierung (S) (WiSe 2018/2019)

Frau Dr. Sina Zarriess
sina.zarriess@uni-bielefeld.de

16. Mai 2019

Inhaltsverzeichnis

1 Einleitung **2**

2 Datengrundlage **3**

3 Methode **4**

3.1 Annotationsschema 5

3.2 Analyse mit XSLT 7

4 Ergebnisse **8**

5 Reflexion **9**

1 Einleitung

E-Mail-Adressen können für non-profit Organisationen ebenso wie für Wirtschaftsunternehmen kostbare Daten darstellen. Über Newsletter – als Form des „E-Mail-Marketing“ im weiteren Sinne – ermöglichen sie eine strategische Außendarstellung durch gezielte *one-to-many* Kommunikation. Im Vergleich zu anderen „Marketinginstrumenten“ fallen dabei geringe Kosten, eine hohe Reichweite sowie umfangreiche Personalisierungsmöglichkeiten ins Auge (vgl. Bucher 2016, S. V). Unmittelbare Reaktionsmöglichkeiten, zum Beispiel das Klicken auf eingebettete Links, machen Mailings insbesondere für den Auf- und Ausbau von Kundenbeziehungen zu einem attraktiven Instrument des Dialogmarketings (vgl. ebd., S. 2). Zudem ermöglichen Öffnungs- und Klickraten, neben weiteren erstellbaren Statistiken, eine detaillierte Auswertung der versendeten Nachrichten (vgl. ebd., S. 100 ff).

Im Kontext eines alltäglichen „Informationsüberflusses“ und der Knappheit an Aufmerksamkeit im digitalen Raum müssen digitale Inhalte nicht nur relevant, sondern auch interessant sein, um wahrgenommen zu werden (vgl. Kumar u. a. 2018, S. 1). Nicht zuletzt die Anpassbarkeit von Mailings sowie zahlreiche Auswertungs- und Optimierungsmöglichkeiten machen für professionell agierende Organisationen und Unternehmen eine Analyse der zurückliegenden Nachrichten unverzichtbar. Im Rahmen eines Praktikums bei der Umwelt- und Menschenrechtsorganisation *urgewald e.V.* wurde für die vorliegende Arbeit der monatliche Newsletter unter „Realbedingungen“ analysiert. Mithilfe der Auszeichnungssprache *Extensible Markup Language* (XML) wurden hierfür spezifische Daten von „Newsletter-Reports“ in eine Datenbank übertragen und anschließend mittels *eXtensible Stylesheet Language Transformations* (XSLT), einer Programmiersprache zur Transformation von XML-Dokumenten, ausgewertet. Das resultierende Dokument, verfasst in der Auszeichnungssprache *Hypertext Markup Language* (HTML), präsentiert die Daten der XML-Datenbank unter Zuhilfenahme eines Webbrowers in verschiedenen (tabellarischen) Formen und fasst die Analyseergebnisse zusammen.

Im Folgenden werden zuerst die zugrunde liegenden Daten der Arbeit und ihr Ursprung näher beschrieben. Das anschließende Kapitel widmet sich den Analysemethoden. Es geht zuerst auf die Annotation und ihr Schema ein, um sodann die einzelnen Schritte der Analyse mittels XSLT und der Transformation in ein HTML-Dokument zu beleuchten. Abschließend folgt neben einer Zusammenfassung der Ergebnisse eine Reflexion des Arbeitsprozesses sowie der Ausgangsdaten und der entstandenen Ergebnisse.

2 Datengrundlage

Der XML-Analyse des vorliegenden Projektberichts liegen Daten aus sogenannten Newsletter-Reports des monatlichen Standard-Mailings der Umwelt- und Menschenrechtsorganisation *urgewald e. V.* zugrunde, die im Zeitraum zwischen Januar 2014 und März 2019 angefallen sind.[1] Anlass für diese Analyse war ein zehnwöchiges Praktikum, das ich von Ende Januar bis Anfang April 2019 im Hauptsitz der Organisation in Sassenberg durchgeführt habe. Ursprünglich als organisationsinterne Auswertung des Newsletters gedacht, habe ich mit dem Einverständnis von urgewald die Daten für dieses Projekt aufbereiten dürfen.

In einem Versandintervall von (idealerweise) einem Monat sendetet urgewald elektronische Rundschreiben via E-Mail zu aktuellen Themen, Kampagnen, Recherchen und Terminen an Personen, die sich schriftlich für den kostenlosen Newsletter angemeldet haben. Eine solche Anmeldung kann sowohl über die Webseite der Organisation stattfinden[2] als auch auf Veranstaltungen oder (Verbraucher-)Messen, auf denen urgewald unter anderem mit dem Ziel vertreten ist, möglichst viele E-Mail-Adressen für die Verteilernachrichten zu sammeln. Die monatlichen Nachrichten werden schließlich über die E-Mail-Marketing-Software CleverReach in einem einheitlichen Format erstellt und an die Mailadressen im Verteiler gesendet.

[1] Vom Standard-Mailing (Newsletter) abweichende Formen des E-Mail-Versands, etwa Standalone-Mailings oder mehrstufige E-Mail-Kampagnen, wurden nicht berücksichtigt.

[2] https://urgewald.org/newsletter

CleverReach erstellt bereits automatisch einige Statistiken zur Analyse der zurückliegenden Mailings. In den Newsletter-Reports der Marketing-Software werden etwa Kennzahlen und Raten zur Zustellung und zu den Empfänger*innen, zu den Öffnungen, Klicks, Bounces und Abmeldungen und nicht zuletzt zur „24 Stunden Performance“ dargestellt. Darüber hinaus bewertet die Software mit einem 5-Punkte-System die Qualität des Betreffs und des Inhalts auf Basis der Öffnungsbeziehungsweise Klickraten. Eine weitere Funktion ist die Möglichkeit eines sogenannten A/B-Tests. Hierbei wird der Newsletter vor dem Versand an den gesamten Verteiler an zwei verschiedene Teilmengen der Abonennt*innen gesendet, die jeweils die aktuelle Nachricht mit einem anderen Betreff (A oder B) empfangen. Dieser Test wird anschließend nach Öffnungen und Klicks ausgewertet, was letztlich entscheidend für das Ergebnis ist, mit welchem Betreff die Software den Newsletter an die verbleibenden Abonennt*innen sendet.

Zur Erstellung der XML-Datenbank habe ich im zeitlichen Verlauf auf verschiedene Inhalte der Newsletter-Reports zurückgegriffen. Den Kern bilden die Öffnungs-, Klick- und Abbestellraten des spezifischen Rundschreibens. Hinzu kommen Angaben zur Sender*in, Empfänger*in, zum Betreff und zur Zustellung. Erweitert habe ich die Datenbank zudem um den A/B-Test, der im Jahre 2018 hinzugekommenen ist. Ab 2019 habe ich ferner auch Teile des Inhalts der Nachrichten in die Annotation integriert, worauf ich in Kapitel 3 näher eingehen werde.

3 Methode

Das Projekt basiert auf der Datenbank `reports.xml`, die in der Auszeichnungssprache XML verfasst ist und welche die hierarchisch strukturierten Daten im Format einer Textdatei darstellt. Zwei Dokumente des Typs *XML Schema Definition* (XSD) definieren als sogenanntes XML Schema zudem die Strukturen eines validen XML-Dokuments: Die Datei `reports.xsd` definiert die grundlegenden Strukturen, die exemplarisch durch die hinzutretende Datei `reports_extension.xsd` ergänzt werden. Das XSLT-Dokument namens `reports.xsl` führt schließlich eine Trans-

formation der XML-Datenbank in die Datei `reports.html` durch, ein Dokument der Auszeichnungssprache HTML5, das mithilfe eines Webbrowers betrachtet werden kann.[1] Folgende Liste fasst die im Projekt beinhalteten Dokumente zusammen:

- `reports.xml` – Datenbank
- `reports.xsd` – grundelegendes XML Schema
- `reports_extension.xsd` – ergänzendes XML Schema
- `reports.xsl` – XML-Transformation
- `reports.html` – Ausgabedatei

3.1 Annotationsschema

Einen Überblick über das verwendete Annotationsschema gibt seine grafische Darstellung als gewurzelter Baum in Abbildung 3.1, die sowohl das XML Schema `reports.xsd` als auch seine Erweiterung `reports_extension.xsd` in ihren Grundstrukturen umfasst. In der Darstellung nicht enthalten sind etwa die in den Element- und Attribut-Deklarationen aufgeführten Spezifikationen möglicher Vorkommen (*occurs*) und der erlaubten Datentypen (*type*), die in der folgenden Beschreibung zum Teil ergänzt werden.

Im grundlegenden Schema `reports.xsd` bildet der Knoten *reports* die Wurzel des Baums, von dem die Tochterknoten *head*, *senderIDs* und *year* abstammen. Aus dem „Dateikopf“ *head* folgen als Blätter des Baumes *update* und *annotation*, von denen *update* mit den Attributen *date* und *name* Informationen zur letzten Aktualisierung der Datenbank bereitstellt und *annotation* mittels *name* und *mail* Auskunft über die Ersteller*in gibt. Als Schwesterknoten von *head* ist *senderIDs* ausschließlich Mutter von Knoten namens *senderID*, die mit den Attributen *sid*, *name*, *mail* und *url* eine referenzierbare ID, den Namen, die E-Mail- sowie eine Webadresse der eingetragenen Absender*in enthalten.

[1] Die Transformation habe ich über die Kommandozeile mittels dem XSLT-Prozessor *xsltproc* mit folgendem Befehl durchgeführt: `xsltproc reports.xsl reports.xml > reports.html`

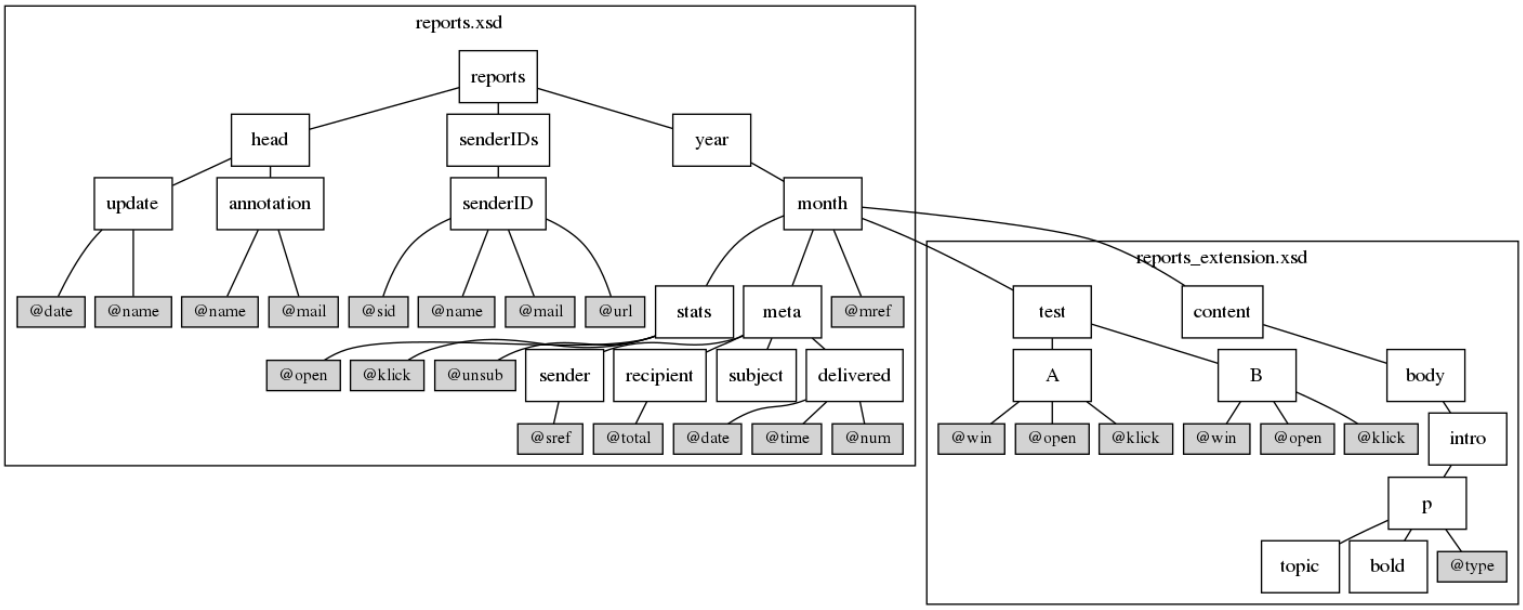

Abbildung 3.1: Annotationsschema

Der „Kern“ der Datenbank ist unter Knoten namens *year* zusammengefasst, die unbegrenzt oft vorkommen können und Daten der monatlichen Newsletter in Jahre gruppieren. Diese enthalten jeweils zwölf Tochterknoten namens *month*, die anhand des Attributs *mref* einem Monat zugeordnet werden. Ein Tochterknoten *stats* enthält mit *open*, *klick* und *unsub* Attribute, die Öffnungs-, Klick- und Abbestellungsraten eines Newsletters umfassen. Daneben entalten Tochterknoten mit dem Namen *meta* (Meta-)Daten zur Absender*in, zu den Empfänger*innen, zum Betreff und zur Zustellung. Dies wird realisiert durch die Knoten *sender* (mit der Referenz durch das Attribut *sref* auf eine ID eines Absenders in *senderID*), *recipient* (mit der Anzahl der Empfäger*innen in dem Attribut *total*), *subject* und *delivered* (mit den Attributen *date*, *time* und *num*, die Angaben zum Datum, zur Uhrzeit und zur Anzahl der tatsächlichen Zustellungen enthalten).

Das zweite Schema `reports_extension.xsd` erweitert die beschriebene Annotation ausgehend vom Knoten *month*, dem es mit *test* und *content* zwei Tochterknoten hinzufügt. Ersterer ergänzt die Datenbank um die Möglichkeit von A/B-Tests und ist Mutter von den Blättern *A* und *B*, die jeweils mittels der Attribute *win*,

open und *klick* Kennzahlen darüber bereitstellen, welche Version „gewonnen“ hat[2] und auf Basis welcher Öffnungs- und Klickraten. Dem gegenüber fügt *content* die Möglichkeit hinzu, Inhalte der Newsletter in die Datenbank miteinzubeziehen; in diesem Fall „Textkörper“ (keine Kopf- und Fußzeilen), die unter Tochterknoten namens *body* eingebunden werden. Von diesen abstammende Knoten namens *intro* umfassen ausschließlich die Einleitung der jeweiligen Newsletter, die in Absätze beziehungsweise Knoten *p* unterteilt sind, deren Funktion durch ein Attribut *type* (mit den Werten „in“, „main“, „topics“, „out“ oder „signature“) beschrieben wird. Schließlich folgen die Blätter *topic* und *bold*, die das jeweilige Hauptthema und Hervorhebungen im Text umfassen.

3.2 Analyse mit XSLT

Die XSLT-Datei der vorliegenden Arbeit transformiert die XML-Datenbank in eine HTML5-Datei. Einleitend möchte ich auf den folgenden Code zu Beginn des Dokuments eingehen, der kenntlich macht, dass es sich um *eXtensible Stylesheet Language Transformations* handelt, die als *output* eine HTML5-Datei ausgeben.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     version="1.0" xmlns:my="http://reports.org">
3 <xsl:output method="html" doctype-system="about:legacy-compat"
     encoding="ISO-8859-1" indent="yes" />
```

Während in der ersten Zeile lediglich festgelegt wird, dass es sich bei der Textdatei um ein XML-Dokument handelt, spezifiziert die zweite Zeile mit dem Attribut *xmlns:xsl* den XSLT-Namensraum sowie das lokale Präfix „xsl“, um Zugang zu dem Namensraum, das heißt zu den Elementen, Attributen und Möglichkeiten von XSLT, zu erhalten. Hinzu kommt mittels des Attributs *xmlns:my* eine weitere Definition eines Präfixes, das Elementen aus dem angegebenen Namensraum „http://reports.org“ vorausgeht, der dem in `reports.xml` erzeugten Namensraum entspricht. Die dritte Zeile macht anschließend kenntlich, dass eine HTML-Datei ausgegeben wird, die aufgrund der Angabe des Attributs *doctype-system* dem Typ

[2] 0 ≔ „verloren“, 1 ≔ „gewonnen“

HTML5 entspricht. Zusammenfassend erzeugt der beschriebene Code die folgenden Zeilen am Anfang der Ausgabedatei (hinzu kommen die schließenden Tags am Ende des Dokuments). Hierbei ist anzumerken, dass das *html*-Tag im weiteren Verlauf der XSLT-Datei explizit angegeben wird und in der Ausgabedatei durch das Namensraum-Attribut lediglich ergänzt wird:

```
1 <!DOCTYPE html>
2 <html xmlns:my="http://reports.org">
```

Das restliche XSLT-Dokument lässt sich grob in vier Teile gliedern. In einem ersten Abschnitt *head* wird im Tag *style* ein internes Stylesheet für die zu erzeugende HTML-Datei bereitgestellt. Darauf folgt der *body*, in dem zuerst ein Inhaltsverzeichnis erzeugt wird, das einen Überblick über die folgenden Inhalte gibt und diese durch *href-Tags* direkt referenziert:

1. Overview
2. Year Overview
3. Sender Analysis
4. Year-Over-Year Analysis (2015–2019)
5. A/B Test Analysis (2018–2019)
6. Opening Analysis (nach dynamischen Schwellenwerten)
7. Weekday Analysis
8. Month Analysis
9. Conclusion

Dieser dritte Teil enthält unter den jeweiligen Überschriften den Code, der Werte aus der XML-Datenbank aufgreift und beispielsweise in tabellarische Darstellungen transformiert. Dazu wird auf Templates zurückgegriffen, die im letzten Teil des Dokuments als direkte Kinder von *stylesheet* angegeben sind.

4 Ergebnisse

Schlussfolgerungen der Analyse sind in dem erzeugten HTML-Dokument unter „Conclusion“ zusammengefasst. „Overview“, „Year Overview“, „Sender Analysis“ und die „Year-Over-Year Analysis“ haben diesbezüglich keine substanziellen Erkenntnisse hervorgebracht, da sie vor allem einen übersichtlichen (chronologisch geordneten) Überblick über die enthaltenen Daten im XML-Dokument geben sollen. Die anschließende „A/B Test Analysis“ hat nahegelegt, warum bestimmte Betreffe mehr „Klicks generieren“ als gegenübergestellte Betreffe, die den A/B-Test „verloren“ haben. So konnte etwa gezeigt werden, dass eine Nachricht mit dem Betreff „Den Sieg zur Niederlage machen, das schafft nur RWE“ (September 2018) öfter geöffnet und geklickt wird als eine identische Nachricht mit dem Betreff „Deka Investment: Raus aus Rüstung und Kohle“. Dies hat mich in Kombination mit weiteren Erkenntnissen der Analyse zu der Annahme verleitet, dass Betreffe möglichst neugierig machende Konstruktionen enthalten sollten, die auf abgenutzte Ausdrücke verzichten sollten[1] und wenn möglich auf RWE Bezug nehmen können.

Die „Opening Analysis“ nach dynamischen Schwellenwerten, die abhängig vom Durchschnitt der Öffnungsraten sind, hat zusätzlich breitere Trends vor Augen führen können, die neben dem Betreff weitere Faktoren in den Blick nehmen, wie den Monat, das Datum, die Uhrzeit oder den Wochentag. Durch die anschließende „Weekday Analysis“ sowie die „Month Analysis“ konnten diese Trends anhängig vom Wochentag beziehungsweise vom Monat explizit betrachtet werden. Insbesondere die Kombination der Daten von verschiedenen Tabellen ist befruchtend und auch notwendig, da dadurch Annahmen erhärtert oder deutlich geschwächt werden können. Beispielsweise verdeutlicht die Analyse nach Monaten, dass Nachrichten zu Beginn des Jahres wahrscheinlich unabhängig vom Betreff in größerer Anzahl geöffnet werden als in späteren Monaten. Darüber hinaus zeigt die Analyse nach Monaten, dass der Newsletter zumeist freitags versendet wird, weshalb etwa ein selteneres oder häufigeres Auftreten von Freitagen (#5) in der „Opening Analysis“ unter „Above-Average“ bezieungsweise „Below-Average“ mit besonderer Signifikanz einhergeht.

[1] „Deka Investment: Raus aus Rüstung und Kohle“ ist ein durchgängiger Kampagnen-Slogan.

5 Reflexion

Ein grundlegendes Problem des Datensatzes besteht darin, dass sich die Statistiken nach der Erhebung weiterhin verändern können, wenn die Empfänger*innen mit den Nachrichten interagieren. Dies wird insbesondere bei relativ aktuellen Monatsmails zum Problem, da sich die Werte in der Regel erst nach ein bis zwei Monaten stabilisieren, sodass sie bis dahin nicht aussagekräftig sind.

Ein weiteres Problem liegt darin begründet, dass sich neben den Kennzahlen auch der Aufbau des Newsletters grundlegend verändern kann, was im Extremfall dazu führen könnte, dass das ergänzende Schema `reports_extension.xsd` unbrauchbar wird, insofern es nicht an die Veränderungen angepasst wird. Da für die vorliegende Analyse nur die Einleitung (*intro*) des Textkörpers (*body*) in das XML Schema integriert wurde, ist das Schema in diesem Fall von einer Umgestaltung des Newsletter-Hauptteils im April 2019 nicht betroffen, bei der neue Elemente wie die „Zahl des Monats“ oder ein Zitatblock hinzugefügt wurden.

Daneben ist auffällig, dass die „Opening Analysis“ zu den überdurchschnittlichen Nachrichten „Above-Average (27.78<)“ auch Nachrichten aus dem Jahre 2014 mit dem einheitlichen Betreff „urgewald e.V. Newsletter“ zählt. Da diese Öffnungsraten auf andere Gründe zurückzuführen sind als auf den Betreff, stellt dies die Aussagekraft der Daten insgesamt infrage. So ist kritisch zu hinterfragen, inwieweit die geschilderten Annahmen der Wahrheit entsprechen oder doch fehlgeleitet sind. Aus diesem Grund sollten im Rahmen einer professionellen Analyse auch statistische Verfahren wie Signifikanztests durchgeführt werden.

Ferner ist insbesondere die „Year-Over-Year Analysis“ durch ineffizienten und sich wiederholenden Code geprägt, der die gleiche Analyse für alle vorkommenden Jahre wiederholt. Dies führt unter anderem dazu, dass das XSLT-Dokument angepasst werden müsste, wenn ein neues Jahr in die Datenbank aufgenommen wird. Einen gekürzten Versuch dies zu umgehen, beschreibe ich im Folgenden, um das zugrunde liegende Problem einer Zusammenfassung des Codes zu erläutern.

```
<xsl:for-each select="my:year/@yref">
        <xsl:call-template name="Year-Over-Year_Analysis">
                <xsl:with-param name="YEAR" select =".”/>
                <xsl:with-param name="COUNT" select ="count(../../
                   my:year[@yref=$YEAR]//my:delivered)"/>
        </xsl:call-template>
</xsl:for-each>
```

```
<xsl:template name="Year-Over-Year_Analysis">
  <xsl:param name="YEAR"/>
  <xsl:param name="COUNT"/>
  <h3><xsl:value-of select="$YEAR"/></h3>
  <xsl:value-of select="$COUNT"/>
</xsl:template>
```

```
$ xsltproc reports.xsl reports.xml > reports.html
runtime error: file reports.xsl line 143 element call-template
Variable 'YEAR' has not been declared.
XPath error : Undefined variable
runtime error: file reports.xsl line 145 element with-param
Failed to evaluate the expression of variable 'COUNT'.
no result for reports.xml
```

Im ersten Ausschnitt des alternativen, jedoch fehlerhaften Ansatzes ist eine *for-each*-Schleife des Codes zu sehen, die ein Template *Year-Over-Year_Analysis* aufruft, das wiederum im zweiten Ausschnitt des Codes zu sehen ist. Der dritte Ausschnitt ist eine Ausgabe des XSLT-Prozessors über die Kommandozeile, die eine Fehlermeldung darstellt. Hierbei wird deutlich, dass der Parameter *COUNT* in seiner Auswahl nicht auf den Wert des Parameters *YEAR* zugreifen kann, der den Wert des aktuellen Jahres annimmt. In der Folge kann der Ausdruck von *COUNT* nicht ausgewertet werden, sodass eine Fehlermeldung ausgegeben wird. Dieser Sachverhalt führt letztlich dazu, dass jedes Jahr einzeln aufgerufen werden muss, damit es in weiteren *select*-Ausdrücken ausgewertet werden kann.

Literatur

Bucher, Martin (2016). *Erfolgreicher Einstieg ins professionelle E-Mail-Marketing.* ger. Hrsg. von Katja Hänsler u. a. Wiesbaden: Springer Fachmedien Wiesbaden. ISBN: 978-3-658-14377-0.

Kumar, Ashish und Jari Salo (2018). „Effects of link placements in email newsletters on their click-through rate". English. In: *Journal of marketing communications, 2018, 24, 5, 535.*

Schriftliche Versicherung

Hiermit versichere ich, dass ich den vorliegenden schriftlichen Projektbericht selbständig angefertigt habe. Alle Stellen, die dem Wortlaut oder dem Sinn nach anderen Werken (ein schließlich elektronischer Quellen) entnommen sind, habe ich in jedem einzelnen Fall unter genauer Angabe der Quelle deutlich gekennzeichnet.

Bielefeld, den 16. Mai 2019

_____

(Max Harder)