Über die Klassifizierung von Knoten in dynamischen Netzwerken mit Inhalt

Martin Thoma

Betreuer: Christopher Oßner

Zusammenfassung In dieser Arbeit wird der DYCOS-Algorithmus, wie er in [AgLi11] vorgestellt wurde, erklärt. Er arbeitet auf Graphen, deren Knoten teilweise mit Beschriftungen versehen sind und ergänzt automatisch Beschriftungen für Knoten, die bisher noch keine Beschriftung haben. Dieser Vorgang wird „Klassifizierung“ genannt. Dazu verwendet er die Struktur des Graphen sowie textuelle Informationen, die den Knoten zugeordnet sind. Die in [AgLi11] beschriebene experimentelle Analyse ergab, dass er auch auf dynamischen Graphen mit 19 396 bzw. 806 635 Knoten, von denen nur 14 814 bzw. 18 999 beschriftet waren, innerhalb von weniger als einer Minute auf einer CPU eines Intel Xeon 2.5GHz mit 32G RAM ausgeführt werden kann.
Zusätzlich wird auf Schwächen von [AgLi11] hingewiesen und mögliche Verbesserungen vorgeschlagen.

Keywords: DYCOS, Label Propagation, Knotenklassifizierung

1 Einleitung

1.1 Motivation

Teilweise gelabelte Netzwerke sind allgegenwärtig. Publikationsdatenbanken mit Publikationen als Knoten, Literaturverweisen und Zitaten als Kanten sowie Tags oder Kategorien als Labels; Wikipedia mit Artikeln als Knoten, Links als Kanten und Kategorien als Labels sowie soziale Netzwerke mit Eigenschaften der Benutzer als Labels sind drei Beispiele dafür. Häufig sind Labels nur teilweise vorhanden und es ist wünschenswert die fehlenden Labels zu ergänzen.

1.2 Problemstellung

Gegeben ist ein Graph, der teilweise gelabelt ist. Zusätzlich stehen zu einer Teilmenge der Knoten Texte bereit. Gesucht sind nun Labels für alle Knoten, die bisher noch nicht gelabelt sind.

Definition 1 (Knotenklassifierungsproblem) *Sei $G_t = (V_t, E_t, V_{L,t})$ ein gerichteter Graph, wobei V_t die Menge aller Knoten, E_t die Kantenmenge und $V_{L,t} \subseteq V_t$ die Menge der gelabelten Knoten jeweils zum Zeitpunkt t bezeichne. Außerdem sei L_t die Menge aller zum Zeitpunkt t vergebenen Labels und $f : V_{L,t} \to L_t$ die Funktion, die einen Knoten auf sein Label abbildet.*

Weiter sei für jeden Knoten $v \in V$ eine (eventuell leere) Textmenge $T(v)$ gegeben.

Gesucht sind nun Labels für $V_t \setminus V_{L,t}$, also $\tilde{f} : V_t \to L_t$ mit $\tilde{f}|_{V_{L,t}} = f$.

1.3 Herausforderungen

Die Graphen, für die dieser Algorithmus konzipiert wurde, sind viele 10 000 Knoten groß und dynamisch. Das bedeutet, es kommen neue Knoten und eventuell auch neue Kanten hinzu bzw. Kanten oder Knoten werden entfernt. Außerdem stehen textuelle Inhalte zu den Knoten bereit, die bei der Klassifikation genutzt werden können. Bei kleinen Modifikationen sollte nicht alles nochmals berechnen werden müssen, sondern basierend auf zuvor berechneten Labels sollte die Klassifizierung modifiziert werden.

2 DYCOS

2.1 Überblick

DYCOS (DYnamic Classification algorithm with cOntent and Structure) ist ein Knotenklassifizierungsalgorithmus, der Ursprünglich in [AgLi11] vorgestellt wurde. Er klassifiziert einzelne Knoten, indem r Random Walks der Länge l, startend bei dem zu klassifizierenden Knoten v gemacht werden. Dabei werden die Labels der besuchten Knoten gezählt. Das Label, das am häufigsten vorgekommen ist, wird als Label für v gewählt. DYCOS nutzt also die sog. Homophilie, d. h. die Eigenschaft, dass Knoten, die nur wenige Hops von einander entfernt sind, häufig auch ähnlich sind [BhCM11]. Der DYCOS-Algorithmus nimmt jedoch nicht einfach den Graphen für dieses Verfahren, sondern erweitert ihn mit Hilfe der zur Verfügung stehenden Texte.
Für diese Erweiterung wird zuerst wird Vokabular W_t bestimmt, das charakteristisch für eine Knotengruppe ist. Wie das gemacht werden kann und warum nicht einfach jedes Wort in das Vokabular aufgenommen wird, wird in Abschnitt 2.4 erläutert.
Nach der Bestimmung des Vokabulars wird für jedes Wort im Vokabular ein Wortknoten zum Graphen hinzugefügt. Alle Knoten, die der Graph zuvor hatte, werden nun „Strukturknoten“ genannt. Ein Strukturknoten v wird genau dann mit einem Wortknoten $w \in W_t$ verbunden, wenn w in einem Text von v vorkommt. Der DYCOS-Algorithmus betrachtet also die Texte, die einem Knoten

zugeordnet sind, als eine Multimenge von Wörtern. Das heißt, zum einen wird nicht auf die Reihenfolge der Wörter geachtet, zum anderen wird bei Texten eines Knotens nicht zwischen verschiedenen Texten unterschieden. Jedoch wird die Anzahl der Vorkommen jedes Wortes berücksichtigt.

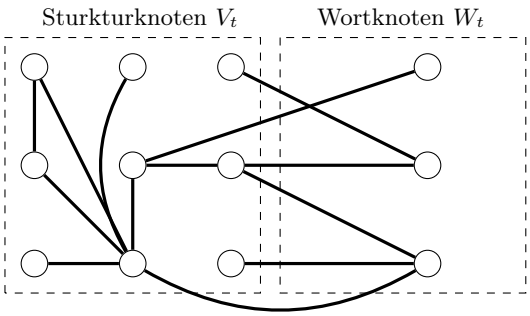


Abbildung 1: Erweiterter Graph

Entsprechend werden zwei unterschiedliche Sprungtypen unterschieden, die strukturellen Sprünge und inhaltliche Mehrfachsprünge:

Definition 2 *Sei $G_{E,t} = (V_t, E_{S,t} \cup E_{W,t}, V_{L,t}, W_t)$ der um die Wortknoten W_t erweiterte Graph.*

*Dann heißt das zufällige wechseln des aktuell betrachteten Knoten $v \in V_t$ zu einem benachbartem Knoten $w \in V_t$ ein **struktureller Sprung**.*

Im Gegensatz dazu benutzten inhaltliche Mehrfachsprünge tatsächlich die Grapherweiterung:

Definition 3 *Sei $G_t = (V_t, E_{S,t} \cup E_{W,t}, V_{L,t}, W_t)$ der um die Wortknoten W_t erweiterte Graph.*

*Dann heißt das zufällige wechseln des aktuell betrachteten Knoten $v \in V_t$ zu einem benachbartem Knoten $w \in W_t$ und weiter zu einem zufälligem Nachbar $v' \in V_t$ von w ein **inhaltlicher Mehrfachsprung**.*

Jeder inhaltliche Mehrfachsprung beginnt und endet also in einem Strukturknoten, springt über einen Wortknoten und ist ein Pfad der Länge 2.

Ob in einem Sprung der Random Walks ein struktureller Sprung oder ein inhaltlicher Mehrfachsprung gemacht wird, wird jedes mal zufällig neu entschieden. Dafür wird der Parameter $0 \leq p_S \leq 1$ für den Algorithmus gewählt. Mit einer Wahrscheinlichkeit von p_S wird eine struktureller Sprung durchgeführt und mit einer Wahrscheinlichkeit von $(1 - p_S)$ ein modifizierter inhaltlicher Mehrfachsprung, wie er in Abschnitt 2.3 erklärt wird, gemacht. Dieser Parameter gibt an, wie wichtig die Struktur des Graphen im Verhältnis zu den textuellen In-

halten ist. Bei $p_S = 0$ werden ausschließlich die Texte betrachtet, bei $p_S = 1$ ausschließlich die Struktur des Graphen.

Die Vokabularbestimmung kann zu jedem Zeitpunkt t durchgeführt werden, muss es aber nicht.

In Algorithmus 1 wird der DYCOS-Algorithmus als Pseudocode vorgestellt.

Algorithmus 1 DYCOS-Algorithmus

Input:
$G_{E,t} = (V_t, E_{S,t} \cup E_{W,t}, V_{L,t}, W_t)$ (Erweiterter Graph),
r (Anzahl der Random Walks),
l (Länge eines Random Walks),
p_s (Wahrscheinlichkeit eines strukturellen Sprungs),
q (Anzahl der betrachteten Knoten in der Clusteranalyse)
Output: Klassifikation von $V_t \setminus V_{L,t}$

for each Knoten v in $V_t \setminus V_{L,t}$ **do**
    $d \leftarrow$ defaultdict
    for i von 1 bis r **do**
        $w \leftarrow v$
        for j von 1 bis l **do**
            $sprungTyp \leftarrow$ RANDOM$(0, 1)$
            if $sprungTyp \leq p_S$ **then**
                $w \leftarrow$ STURKTURELLERSPRUNG(w)
            else
                $w \leftarrow$ INHALTLICHERMEHRFACHSPRUNG(w)
            $w \leftarrow v$.GETLABEL() ▷ Zähle das Label
            $d[w] \leftarrow d[w] + 1$
    if d ist leer **then** ▷ Es wurde kein gelabelter Knoten gesehen
        $M_H \leftarrow$ HÄUFIGSTELABELIMGRAPH()
    else
        $M_H \leftarrow$ MAX(d)

    //Wähle aus der Menge der häufigsten Label M_H zufällig eines aus
    $label \leftarrow$ RANDOM(M_H)
    v.ADDLABEL($label$) ▷ und weise dieses v zu
return Labels für $V_t \setminus V_{L,t}$

2.2 Datenstrukturen

Zusätzlich zu dem gerichteten Graphen $G_t = (V_t, E_t, V_{L,t})$ verwaltet der DYCOS-Algorithmus zwei weitere Datenstrukturen:

- Für jeden Knoten $v \in V_t$ werden die vorkommenden Wörter, die auch im Vokabular W_t sind, und deren Anzahl gespeichert. Das könnte z. B. über ein assoziatives Array geschehen. Wörter, die nicht in Texten von v vorkommen, sind nicht im Array. Für alle vorkommenden Wörter ist der gespeicherte Wert zum Schlüssel „Wort“ die Anzahl der Vorkommen von „Wort“ in den Texten von v.
- Für jedes Wort des Vokabulars W_t wird eine Liste von Knoten verwaltet, in deren Texten das Wort vorkommt.
- An einigen Stellen macht ein assoziatives Array, auch „dictionary“ oder „map“ genannt, Sinn. Zustätzlich ist es nützlich, wenn diese Datenstruktur für unbekannte Schlüssel keinen Fehler ausgibt, sondern für diese Schlüssel den Wert 0 annimmt. Eine solche Datenstruktur wird in Python `defaultdict` genannt und ich werde im Folgenden diese Benennung beibehalten.

2.3 Sprungtypen

Die beiden bereits definierten Sprungtypen, der strukturelle Sprung sowie der inhaltliche Mehrfachsprung werden im folgenden erklärt.

Der strukturelle Sprung entspricht einer zufälligen Wahl eines Nachbarknotens, wie es in Algorithmus 2 gezeigt wird.

Algorithmus 2 Struktureller Sprung

```
1: function StrukturellerSprung(Knoten v, Anzahl q)
2:     n ← v.NeighborCount          ▷ Wähle aus der Liste der Nachbarknoten
3:     r ← RandomInt(0, n − 1)                          ▷ einen zufällig aus
4:     v ← v.Next(r)                                    ▷ Gehe zu diesem Knoten
5:     return v
```

Bei inhaltlichen Mehrfachsprüngen ist jedoch nicht sinnvoll so direkt nach der Definition vorzugehen, also direkt von einem strukturellem Knoten $v \in V_t$ zu einem mit v verbundenen Wortknoten $w \in W_t$ zu springen und von diesem wieder zu einem verbundenem strukturellem Knoten $v' \in V_t$. Würde man dies machen, wäre zu befürchten, dass aufgrund von Homonymen die Qualität der Klassifizierung verringert wird. So hat „Brücke“ im Deutschen viele Bedeutungen. Gemeint sein können z. B. das Bauwerk, das Entwurfsmuster der objektorientierten Programmierung oder ein Teil des Gehirns.

Deshalb wird für jeden Knoten v, von dem aus man einen inhaltlichen Mehrfachsprung machen will folgendes Clusteranalyse durchgeführt:

C1 Gehe alle in v startenden Random Walks der Länge 2 durch und erstelle eine Liste L, der erreichbaren Knoten v'. Speichere außerdem, durch wie viele Pfade diese Knoten v' jeweils erreichbar sind.

C2 Betrachte im folgenden nur die Top-q Knoten, wobei $q \in \mathbb{N}$ eine zu wählende Konstante des Algorithmus ist.[1]

C3 Wähle mit Wahrscheinlichkeit $\frac{\textsc{Anzahl}(v')}{\sum_{w \in L} \textsc{Anzahl}(v')}$ den Knoten v' als Ziel des Mehrfachsprungs.

Konkret könnte also ein Inhaltlicher Mehrfachsprung sowie wie in Algorithmus 3 beschrieben umgesetzt werden.

Algorithmus 3 Inhaltlicher Mehrfachsprung

```
1: function InhaltlicherMehrfachsprung(Knoten v)
2:     //Alle Knoten bestimmen, die von v aus über Pfade der Länge 2 erreichbar sind
3:     //Zusätzlich wird für diese Knoten die Anzahl der Pfade der Länge 2 bestimmt,
4:     //durch die sie erreichbar sind
5:     reachableNodes ← defaultdict
6:     for each Wortknoten w in v.getWordNodes() do
7:         for each Strukturknoten x in w.getStructuralNodes() do
8:             reachableNodes[x] ← reachableNodes[x] + 1
9:     //Im folgenden wird davon ausgegangen, dass man über Indizes wahlfrei auf
10:    //Elemente aus M_H zugreifen kann. Dies muss bei der konkreten Wahl
11:    //der Datenstruktur berücksichtigt werden.
12:    M_H ← max(reachableNodes, q)  ▷ Also: |M_H| = q, falls |reachableNodes| ≥ q
13:    //Dictionary mit relativen Häufigkeiten erzeugen
14:    s ← 0
15:    for each Knoten x in M_H do
16:        s ← s + reachableNodes[x]
17:    relativeFrequency ← Dictionary
18:    for each Knoten x in M_H do
19:        relativeFrequency ← reachableNodes[x] / s
20:    //Wähle Knoten i mit einer Wahrscheinlichkeit entsprechend seiner relativen
21:    //Häufigkeit an Pfaden der Länge 2
22:    random ← random(0, 1)
23:    r ← 0.0
24:    i ← 0
25:    while s < random do
26:        r ← r + relativeFrequency[i]
27:        i ← i + 1
28:    v ← M_H[i − 1]
29:    return v
```

[1] Sowohl für den DBLP, als auch für den CORA-Datensatz wurde in [AgLi11, S. 364] $q = 10$ gewählt.

2.4 Vokabularbestimmung

Da die Größe des Vokabulars die Datenmenge signifikant beeinflusst, liegt es in unserem Interesse so wenig Wörter wie möglich ins Vokabular aufzunehmen. Insbesondere sind Wörter nicht von Interesse, die in fast allen Texten vorkommen, wie im Deutschen z. B. „und“, „mit“ und die Pronomen. Es ist wünschenswert Wörter zu wählen, die die Texte möglichst stark voneinander Unterscheiden. Der DYCOS-Algorithmus wählt die Top-m dieser Wörter als Vokabular, wobei $m \in \mathbb{N}$ eine Festzulegende Konstante ist. In [AgLi11, S. 365] wird der Einfluss von $m \in \{ 5, 10, 15, 20 \}$ auf die Klassifikationsgüte untersucht und festgestellt, dass die Klassifikationsgüte mit größerem m sinkt, sie also für $m = 5$ für den DBLP-Datensatz am höchsten ist. Für den CORA-Datensatz wurde mit $m \in \{ 3, 4, 5, 6 \}$ getestet und kein signifikanter Unterschied festgestellt.

Nun kann man manuell eine Liste von zu beachtenden Wörtern erstellen oder mit Hilfe des Gini-Koeffizienten automatisch ein Vokabular erstellen. Der Gini-Koeffizient ist ein statistisches Maß, das die Ungleichverteilung bewertet. Er ist immer im Intervall $[0, 1]$, wobei 0 einer Gleichverteilung entspricht und 1 der größtmöglichen Ungleichverteilung.

Sei nun $n_i(w)$ die Häufigkeit des Wortes w in allen Texten mit dem i-ten Label.

$$p_i(w) := \frac{n_i(w)}{\sum_{j=1}^{|\mathcal{L}_t|} n_j(w)} \qquad \text{(Relative Häufigkeit des Wortes } w\text{)} \qquad (1)$$

$$G(w) := \sum_{j=1}^{|\mathcal{L}_t|} p_j(w)^2 \qquad \text{(Gini-Koeffizient von } w\text{)} \qquad (2)$$

In diesem Fall ist $G(w) = 0$ nicht möglich, da zur Vokabularbestimmung nur Wörter betrachtet werden, die auch vorkommen.

Ein Vorschlag, wie die Vokabularbestimmung implementiert werden kann, ist als Pseudocode mit Algorithmus 4 gegeben. Dieser Algorithmus benötigt neben dem Speicher für den Graphen, die Texte sowie die m Vokabeln noch $\mathcal{O}(|\text{Verschiedene Wörter in } S_t| \cdot (|\mathcal{L}_t| + 1))$ Speicher. Die Average-Case Zeitkomplexität beträgt $\mathcal{O}(|\text{Wörter in } S_t|)$, wobei dazu die Vereinigung von Mengen M, N in $\mathcal{O}(\min |M|, |N|)$ sein muss.

Die Menge S_t kann durch Aus der Menge aller Dokumenten, deren Knoten gelabelt sind, mithilfe des in [Vitt85] vorgestellten Algorithmus bestimmt werden.

3 Schwächen und Verbesserungsvorschläge

Der in [AgLi11] vorgestellte Algorithmus hat einige Probleme, die im Folgenden erläutert werden. Außerdem werden Verbesserungen vorgeschlagen, die es allerdings noch zu untersuchen gilt.

Algorithmus 4 Vokabularbestimmung

Input:
$V_{L,t}$ (Knoten mit Labels),
$\mathcal{L}_t$ (Labels),
$f : V_{L,t} \to \mathcal{L}_t$ (Label-Funktion),
m (Gewünschte Vokabulargröße)
Output: $\mathcal{M}_t$ (Vokabular)

$S_t \leftarrow \textsc{Sample}(V_{L,t})$ ▷ Wähle eine Teilmenge $S_t \subseteq V_{L,t}$ aus
$\mathcal{M}_t \leftarrow \bigcup_{v \in S_t} \textsc{getTextAsSet}(v)$ ▷ Menge aller Wörter
$cLabelWords \leftarrow (|\mathcal{L}_t| + 1) \times |\mathcal{M}_t|$-Array, mit 0en initialisiert

for each $v \in V_{L,t}$ **do** ▷ Gehe jeden Text Wort für Wort durch
    $i \leftarrow \textsc{getLabel}(v)$
    for each $(word, occurences) \in \textsc{getTextAsMultiset}(v)$ **do**
        $cLabelWords[i][word] \leftarrow cLabelWords[i][word] + occurences$
        $cLabelWords[i][|\mathcal{L}_t|] \leftarrow cLabelWords[i][|\mathcal{L}_t|] + occurences$

for each Wort $w \in \mathcal{M}_t$ **do**
    $p \leftarrow$ Array aus $|\mathcal{L}_t|$ Zahlen in $[0, 1]$
    for each Label $i \in \mathcal{L}_t$ **do**
        $p[i] \leftarrow \frac{cLabelWords[i][w]}{cLabelWords[i][|\mathcal{L}_t|]}$
    w.gini $\leftarrow \textsc{sum}(\textsc{map}(\textsc{square}, p))$
$\mathcal{M}_t \leftarrow \textsc{SortDescendingByGini}(\mathcal{M}_t)$
return $\textsc{Top}(\mathcal{M}_t, m)$

3.1 Schwächen von DYCOS

3.1.1 Anzahl der Labels So, wie der DYCOS-Algorithmus vorgestellt wurde, können nur Graphen bearbeitet werden, deren Knoten höchstens ein Label haben. In vielen Fällen, wie z. B. Wikipedia mit Kategorien als Labels haben Knoten jedoch viele Labels.

Auf einen ersten Blick ist diese Schwäche einfach zu beheben, indem man beim zählen der Labels für jeden Knoten jedes Label zählt. Dann wäre noch die Frage zu klären, mit wie vielen Labels der betrachtete Knoten gelabelt werden soll.

Jedoch ist z. B. bei Wikipedia-Artikeln auf den Knoten eine Hierarchie definiert. So ist die Kategorie „Klassifikationsverfahren“ eine Unterkategorie von „Klassifikation“. Bei dem Kategorisieren von Artikeln sind möglichst spezifische Kategorien vorzuziehen, also kann man nicht einfach bei dem Auftreten der Kategorie „Klassifikationsverfahren“ sowohl für diese Kategorie als auch für die Kategorie „Klassifikation“ zählen.

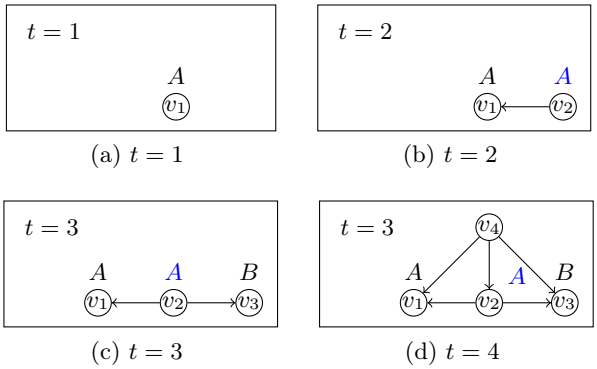
3.1.2 Überanpassung und Reklassifizierung Aggarwal und Li beschreiben in [AgLi11] nicht, auf welche Knoten der Klassifizierungsalgorithmus angewendet werden soll. Jedoch ist die Reihenfolge der Klassifizierung relevant. Dazu folgendes Minimalbeispiel:

Gegeben sei ein dynamischer Graph ohne textuelle Inhalte. Zum Zeitpunkt $t = 1$ habe dieser Graph genau einen Knoten v_1 und v_1 sei mit dem Label A beschriftet. Zum Zeitpunkt $t = 2$ komme ein nicht-gelabelter Knoten v_2 sowie die Kante (v_2, v_1) hinzu.
Nun wird der DYCOS-Algorithmus auf diesen Knoten angewendet und v_2 mit A gelabelt.
Zum Zeitpunkt $t = 3$ komme ein Knoten v_3, der mit B gelabelt ist, und die Kante (v_2, v_3) hinzu.

Würde man nun den DYCOS-Algorithmus erst jetzt, also anstelle von Zeitpunkt $t = 2$ zum Zeitpunkt $t = 3$ auf den Knoten v_2 anwenden, so würde eine 50%-Wahrscheinlichkeit bestehen, dass dieser mit B gelabelt wird. Aber in diesem Beispiel wurde der Knoten schon zum Zeitpunkt $t = 2$ gelabelt. Obwohl es in diesem kleinem Beispiel noch keine Rolle spielt, wird das Problem klar, wenn man weitere Knoten einfügt:

Wird zum Zeitpunkt $t = 4$ ein ungelabelter Knoten v_4 und die Kanten (v_1, v_4), (v_2, v_4), (v_3, v_4) hinzugefügt, so ist die Wahrscheinlichkeit, dass v_4 mit A gelabelt wird bei $\frac{2}{3}$. Werden die als ungelabelten Knoten jedoch erst jetzt und alle gemeinsam gelabelt, so ist die Wahrscheinlichkeit für A als Label bei nur 50%. Bei dem DYCOS-Algorithmus findet also eine Überanpassung an vergangene Labels statt.

Das Reklassifizieren von Knoten könnte eine mögliche Lösung für dieses Problem sein. Knoten, die durch den DYCOS-Algorithmus gelabelt wurden könnten ei-



(a) $t = 1$ (b) $t = 2$ (c) $t = 3$ (d) $t = 4$

Abbildung 2: Minimalbeispiel für den Einfluss früherer DYCOS-Anwendungen

ne Lebenszeit bekommen (TTL, Time to Live). Ist diese abgelaufen, wird der DYCOS-Algorithmus erneut auf den Knoten angewendet.

3.2 Schwächen des Papers

In [AgLi11] wurde eine experimentelle Analyse mithilfe des DBLP-Datensatzes[2] und des CORA-Datensatzes[3] durchgeführt. Die Ergebnisse dieser Analyse können aus folgenden Gründen nicht überprüft werden:

- Der Parameter $a \in \mathbb{N}$, der die Anzahl der ausgehenden Kanten aller Wortknoten beschränkt, wird erst mit der Experimentellen Analyse auf S. 362 eingeführt. Es ist nicht klar, wie entschieden wird welche Kanten gespeichert werden und welche nicht.

- Für die Analyse der CORA-Datensatzes analysiert. Dieser beinhaltet Forschungsarbeiten, wobei die Forschungsgebiete die in einen Baum mit 73 Blättern eingeordnet wurden. Aus diesen 73 Blättern wurden 5 Klassen extrahiert und der Graph, der keine Zeitpunkte beinhaltet, künstlich in 10 Graphen mit Zeitpunkten unterteilt. Wie jedoch diese Unterteilung genau durchgeführt wurde kann nicht nachvollzogen werden.

- Der auf S. 360 in „Algorithm 1“ vorgestellte Pseudocode soll den DYCOS-Algorithmus darstellen. Allerdings werden die bereits klassifizierten Knoten $\mathcal{T}_t$ neu klassifiziert und mit θ die Klassifikationsgüte gemessen.

[2] http://dblp.uni-trier.de/
[3] http://people.cs.umass.edu/ mccallum/data/cora-classify.tar.gz

4 Ausblick

Den sehr einfach aufgebauten DYCOS-Algorithmus kann man noch an vielen Punkten verbessern. So könnte man vor der Auswahl des Vokabulars jedes Wort auf den Wortstamm zurückführen. Dafür könnte zum Beispiel der in [Port97] vorgestellte Porter-Stemming-Algorithmus verwendet werden. Durch diese Maßnahme wird das Vokabular kleiner gehalten, mehr Artikel können mit einander durch Vokabular verbunden werden und der Gini-Koeffizient wird ein besseres Maß für die Gleichheit von Texten.

Eine weitere Verbesserungsmöglichkeit besteht in der Textanalyse. Momentan ist diese noch sehr einfach gestrickt und ignoriert die Reihenfolge von Wortern beziehungsweise Wertungen davon. So könnte man den DYCOS-Algorithmus in einem sozialem Netzwerk verwenden wollen, in dem politische Parteiaffinität von einigen Mitgliedern angegeben wird um die Parteiaffinität der restlichen Mitglieder zu bestimmen. In diesem Fall macht es jedoch einen wichtigen Unterschied, ob jemand über eine Partei gutes oder schlechtes schreibt.

Eine einfache Erweiterung des DYCOS-Algorithmus wäre der Umgang mit mehreren Labels.

DYCOS beschränkt sich bei inhaltlichen Mehrfachsprüngen auf die Top-q-Wortknoten, also die q ähnlichsten Knoten gemessen mit der Aggregatanalyse, allerdings wurde bisher noch nicht untersucht, wie der Einfluss von $q \in \mathbb{N}$ auf die Klassifikationsgüte ist.

Literatur

AgLi11. Charu C. Aggarwal und Nan Li. On Node Classification in Dynamic Content-based Networks. In *SDM*, 2011, S. 355–366.

BhCM11. Smriti Bhagat, Graham Cormode und S. Muthukrishnan. Node Classification in Social Networks, 2011.

Port97. M. F. Porter. Readings in Information Retrieval. Kapitel An Algorithm for Suffix Stripping, S. 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

Vitt85. Jeffrey S. Vitter. Random Sampling with a Reservoir. *ACM Trans. Math. Softw.*, 11(1), 1985, S. 37–57.