

# Über die Klassifizierung von Knoten in dynamischen Netzwerken mit Inhalt

Martin Thoma

Betreuer: Christopher Ofner

**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 einem Kern einer Intel Xeon 2.5GHz CPU mit 32G RAM ausgeführt werden kann.

Zusätzlich wird [AgLi11] kritisch Erörtert und es werden mögliche Erweiterungen des DYCOS-Algorithmus vorgeschlagen.

**Keywords:** DYCOS, Label Propagation, Knotenklassifizierung

## 1 Einleitung

Im Folgenden werden in Abschnitt 1.1 einige Beispiele, in denen der DYCOS-Algorithmus anwendung finden könnte, dargelegt. In Abschnitt 1.2 wird die Problemstellung formal definiert und in Abschnitt 1.3 wird auf besondere Herausforderungen der Aufgabenstellung hingewiesen.

### 1.1 Motivation

Teilweise beschriftete Graphen sind allgegenwärtig. Publikationsdatenbanken mit Publikationen als Knoten, Literaturverweisen und Zitaten als Kanten sowie von Nutzern vergebene Beschriftungen (sog. *Tags*) oder Kategorien als Knotenbeschriftungen; Wikipedia mit Artikeln als Knoten, Links als Kanten und Kategorien als Knotenbeschriftungen sowie soziale Netzwerke mit Eigenschaften der Benutzer als Knotenbeschriftungen sind drei Beispiele dafür. Häufig sind Knotenbeschriftungen nur teilweise vorhanden und es ist wünschenswert die fehlenden Knotenbeschriftungen automatisiert zu ergänzen.

## 1.2 Problemstellung

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

**Definition 1 (Knotenklassifizierungsproblem)** 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 beschrifteten Knoten jeweils zum Zeitpunkt  $t$  bezeichne. Außerdem sei  $L_t$  die Menge aller zum Zeitpunkt  $t$  vergebenen Knotenbeschriftungen und  $f : V_{L,t} \rightarrow L_t$  die Funktion, die einen Knoten auf seine Beschriftung abbildet.

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

Gesucht sind nun Beschriftungen für  $V_t \setminus V_{L,t}$ , also  $\tilde{f} : V_t \rightarrow L_t$  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. „Dynamisch“ bedeutet in diesem Kontext, dass neue Knoten und eventuell auch neue Kanten hinzu kommen bzw. Kanten oder Knoten entfernt werden. Außerdem stehen textuelle Inhalte zu den Knoten bereit, die bei der Klassifikation genutzt werden können. Bei kleinen Änderungen sollte nicht alles nochmals berechnet werden müssen, sondern basierend auf zuvor berechneten Knotenbeschriftungen sollte die Klassifizierung angepasst werden.

## 2 DYCOS

### 2.1 Überblick

DYCOS (DYnamic Classification algorithm with cOntent and Structure) ist ein Knotenklassifizierungsalgorithmus, der Ursprünglich in [AgLi11] vorgestellt wurde.

Ein zentrales Element des DYCOS-Algorithmus ist der sog. *Random Walk*:

**Definition 2 (Random Walk, Sprung)** Sei  $G = (V, E)$  mit  $E \subseteq V \times V$  ein Graph und  $v_0 \in V$  ein Knoten des Graphen.

Ein *Random Walk* der Länge  $l$  auf  $G$ , startend bei  $v_0$  ist nun der zeitdiskrete stochastische Prozess, der  $v_i$  auf einen zufällig gewählten Nachbarn  $v_{i+1}$  abbildet (für  $i \in 0, \dots, l-1$ ). Die Abbildung  $v_i \mapsto v_{i+1}$  heißt ein *Sprung*.

Der DYCOS-Algorithmus klassifiziert einzelne Knoten, indem  $r$  Random Walks der Länge  $l$ , startend bei dem zu klassifizierenden Knoten  $v$  gemacht werden. Dabei werden die Beschriftungen der besuchten Knoten gezählt. Die Beschriftung, die am häufigsten vorgekommen ist, wird als Beschriftung 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 arbeitet jedoch nicht direkt auf dem Graphen, sondern erweitert ihn mit Hilfe der zur Verfügung stehenden Texte. Wie diese Erweiterung erstellt wird, wird im Folgenden erklärt.

Für diese Erweiterung wird zuerst das 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.

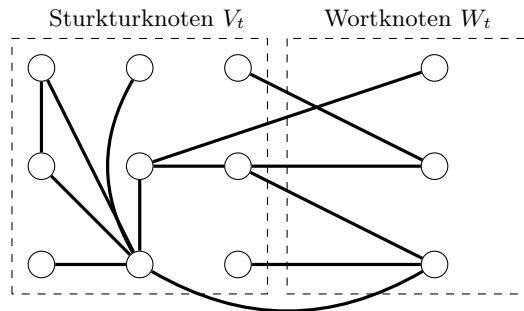


Abbildung 1: Erweiterter Graph

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

**Definition 3 (struktureller Sprung)** 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 4 (inhaltlicher Mehrfachsprung)** 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 benachbarten 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 ein struktureller Sprung durchgeführt und mit einer Wahrscheinlichkeit von  $(1 - p_S)$  ein modifizierter inhaltlicher Mehrfachsprung, wie er in Abschnitt 2.3 erklärt wird, gemacht. Der Parameter  $p_S$  gibt an, wie wichtig die Struktur des Graphen im Verhältnis zu den textuellen Inhalten 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.

## 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  $w \in W_t$  die Anzahl der Vorkommen von  $w$  in den Texten von  $v$ .
- Für jedes Wort des Vokabulars  $W_t$  wird eine Liste von Knoten verwaltet, in deren Texten das Wort vorkommt. Diese Liste wird bei den inhaltlichen Mehrfachsprung, der in Abschnitt 2.3 erklärt wird, verwendet.
- An einigen Stellen macht ein assoziatives Array, auch „dictionary“ oder „map“ genannt, Sinn. Zusätzlich 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.

---

**Algorithmus 1** DYCOS-Algorithmus

---

**Input:**

- 1:  $G_{E,t} = (V_t, E_{S,t} \cup E_{W,t}, V_{L,t}, W_t)$  (Erweiterter Graph),
- 2:  $r$  (Anzahl der Random Walks),
- 3:  $l$  (Länge eines Random Walks),
- 4:  $p_s$  (Wahrscheinlichkeit eines strukturellen Sprungs),
- 5:  $q$  (Anzahl der betrachteten Knoten in der Clusteranalyse)

**Output:** Klassifikation von  $V_t \setminus V_{L,t}$ 

```

6:
7:
8: for each Knoten  $v \in V_t \setminus V_{L,t}$  do
9:    $d \leftarrow$  defaultdict
10:  for  $i = 1, \dots, r$  do
11:     $w \leftarrow v$ 
12:    for  $j = 1, \dots, l$  do
13:       $sprungTyp \leftarrow$  RANDOM(0, 1)
14:      if  $sprungTyp \leq p_s$  then
15:         $w \leftarrow$  STURKTURELLERSPRUNG( $w$ )
16:      else
17:         $w \leftarrow$  INHALTLICHERMEHRFACHSPRUNG( $w$ )
18:         $w \leftarrow v.GETLABEL()$  ▷ Zähle die Beschriftung
19:         $d[w] \leftarrow d[w] + 1$ 
20:      if  $d$  ist leer then ▷ Es wurde kein beschrifteter Knoten gesehen
21:         $M_H \leftarrow$  HÄUFIGSTELABELIMGRAPH()
22:      else
23:         $M_H \leftarrow$  MAX( $d$ )
24:
25:      //Wähle aus der Menge der häufigsten Beschriftungen  $M_H$  zufällig eine aus
26:       $label \leftarrow$  RANDOM( $M_H$ )
27:       $v.ADDLABEL(label)$  ▷ und weise dieses  $v$  zu
28: return Beschriftungen für  $V_t \setminus V_{L,t}$ 

```

---

### 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 STURKTURELLERSPRUNG(Knoten  $v$ , Anzahl  $q$ )
2:    $n \leftarrow v$ .NEIGHBORCOUNT           ▷ Wähle aus der Liste der Nachbarknoten
3:    $r \leftarrow \text{RANDOMINT}(0, n - 1)$        ▷ einen zufällig aus
4:    $v \leftarrow v$ .NEXT( $r$ )                 ▷ Gehe zu diesem Knoten
5:   return  $v$ 

```

---

Bei inhaltlichen Mehrfachsprüngen ist jedoch nicht sinnvoll so strikt 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 folgende Textanalyse 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 bzgl. der Anzahl der Pfade von  $v$  nach  $v'$ , wobei  $q \in \mathbb{N}$  eine zu wählende Konstante des DYCOS-Algorithmus ist.<sup>1</sup> Diese Knotenmenge heiße im Folgenden  $T(v)$  und  $p(v, v')$  sei die Anzahl der Pfade von  $v$  über einen Wortknoten nach  $v'$ .
- C3 Wähle mit Wahrscheinlichkeit  $\frac{p(v, v')}{\sum_{w \in T(v)} p(v, w)}$  den Knoten  $v' \in T(v)$  als Ziel des Mehrfachsprungs.

Konkret könnte also ein inhaltlicher Mehrfachsprung sowie wie in Algorithmus 3 beschrieben umgesetzt werden. Der Algorithmus bekommt einen Startknoten  $v \in V_T$  und einen  $q \in \mathbb{N}$  als Parameter.  $q$  ist ein Parameter der für den DYCOS-Algorithmus zu wählen ist. Dieser Parameter beschränkt die Anzahl der mögli-

---

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

chen Zielknoten  $v' \in V_T$  auf diejenigen  $q$  Knoten, die  $v$  bzgl. der Textanalyse am ähnlichsten sind.

In Zeile 2 bis Zeile 5 wird Punkt C1 durchgeführt.

In Zeile 6 wird Punkt C2 durchgeführt. Bei der Wahl der Datenstruktur von  $T$  ist zu beachten, dass man in Zeile 21 über Indizes auf Elemente aus  $T$  zugreifen können muss.

In Zeile 8 bis Zeile 13 wird ein Wörterbuch erstellt, das von  $v' \in T(v)$  auf die relative Häufigkeit bzgl. aller Pfade von  $v$  zu Knoten aus den Top- $q$  abbildet.

In allen folgenden Zeilen wird Punkt C3 durchgeführt. In Zeile 15 bis Zeile 22 wird ein Knoten  $v' \in T(v)$  mit einer Wahrscheinlichkeit, die seiner relativen Häufigkeit am Anteil der Pfade der Länge 2 von  $v$  nach  $v'$  über einen beliebigen Wortknoten entspricht ausgewählt und schließlich zurückgegeben.

---

### Algorithmus 3 Inhaltlicher Mehrfachsprung

---

```

1: function INHALTLICHERMEHRFACHSPRUNG(Knoten  $v \in V_T$ ,  $q \in \mathbb{N}$ )
2:    $reachableNodes \leftarrow$  defaultdict
3:   for each Wortknoten  $w$  in  $v.GETWORDNODES()$  do
4:     for each Strukturknoten  $x$  in  $w.GETSTRUCTURALNODES()$  do
5:        $reachableNodes[x] \leftarrow reachableNodes[x] + 1$ 
6:    $T \leftarrow \text{MAX}(reachableNodes, q)$      $\triangleright$  Also:  $|T| = q$ , falls  $|reachableNodes| \geq q$ 
7:
8:    $s \leftarrow 0$ 
9:   for each Knoten  $x \in T$  do
10:     $s \leftarrow s + reachableNodes[x]$ 
11:   $relativeFrequency \leftarrow$  Dictionary
12:  for each Knoten  $x \in T$  do
13:     $relativeFrequency \leftarrow \frac{reachableNodes[x]}{s}$ 
14:
15:   $random \leftarrow \text{RANDOM}(0, 1)$ 
16:   $r \leftarrow 0.0$ 
17:   $i \leftarrow 0$ 
18:  while  $s < random$  do
19:     $r \leftarrow r + relativeFrequency[i]$ 
20:     $i \leftarrow i + 1$ 
21:   $v \leftarrow T[i - 1]$ 
22:  return  $v$ 

```

---

## 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 der  $i$ -ten Knotenbeschriftung.

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

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

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  $\mathcal{O}(|\text{Verschiedene Wörter in } S_t| \cdot (|\mathcal{L}_i| + 1))$  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 aus der Menge aller Dokumente, deren Knoten beschriftet 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.

#### 3.1 Anzahl der Knotenbeschriftungen

So, wie der DYCOS-Algorithmus vorgestellt wurde, können nur Graphen bearbeitet werden, deren Knoten jeweils höchstens eine Beschriftung haben. In vie-



---

**Algorithmus 4** Vokabularbestimmung

---

**Input:**

- 1:  $V_{L,t}$  (beschriftete Knoten),
- 2:  $\mathcal{L}_t$  (Beschriftungen),
- 3:  $f : V_{L,t} \rightarrow \mathcal{L}_t$  (Beschriftungsfunktion),
- 4:  $m$  (Gewünschte Vokabulargröße)

**Output:**  $\mathcal{M}_t$  (Vokabular)

- 5:
  - 6:  $S_t \leftarrow \text{SAMPLE}(V_{L,t})$  ▷ Wähle eine Teilmenge  $S_t \subseteq V_{L,t}$  aus
  - 7:  $\mathcal{M}_t \leftarrow \bigcup_{v \in S_t} \text{GETTEXTASSET}(v)$  ▷ Menge aller Wörter
  - 8:  $cLabelWords \leftarrow (|\mathcal{L}_t| + 1) \times |\mathcal{M}_t|$ -Array, mit 0en initialisiert
  - 9:
  - 10: **for each**  $v \in V_{L,t}$  **do** ▷ Gehe jeden Text Wort für Wort durch
  - 11:      $i \leftarrow \text{GETLABEL}(v)$
  - 12:     **for each**  $(word, occurrences) \in \text{GETTEXTASMULTISET}(v)$  **do**
  - 13:          $cLabelWords[i][word] \leftarrow cLabelWords[i][word] + occurrences$
  - 14:          $cLabelWords[i][|\mathcal{L}_t|] \leftarrow cLabelWords[i][|\mathcal{L}_t|] + occurrences$
  - 15:
  - 16: **for each** Wort  $w \in \mathcal{M}_t$  **do**
  - 17:      $p \leftarrow$  Array aus  $|\mathcal{L}_t|$  Zahlen in  $[0, 1]$
  - 18:     **for each** Label  $i \in \mathcal{L}_t$  **do**
  - 19:          $p[i] \leftarrow \frac{cLabelWords[i][w]}{cLabelWords[i][|\mathcal{L}_t|]}$
  - 20:      $w.gini \leftarrow 0$
  - 21:     **for each**  $i \in 1, \dots, |\mathcal{L}_t|$  **do**
  - 22:          $w.gini \leftarrow w.gini + p[i]^2$
  - 23:  $\mathcal{M}_t \leftarrow \text{SORTDESCENDINGBYGINI}(\mathcal{M}_t)$
  - 24: **return**  $\text{TOP}(\mathcal{M}_t, m)$
-

len Fällen, wie z. B. Wikipedia mit Kategorien als Knotenbeschriftungen haben Knoten jedoch viele Beschriftungen.

Auf einen ersten Blick ist diese Schwäche einfach zu beheben, indem man beim zählen der Knotenbeschriftungen für jeden Knoten jedes Beschriftung zählt. Dann wäre noch die Frage zu klären, mit wie vielen Beschriftungen der betrachtete Knoten beschriftet 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.

### 3.2 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  $A$  beschriftet. Zum Zeitpunkt  $t = 2$  komme ein nicht beschrifteter Knoten  $v_2$  sowie die Kante  $(v_2, v_1)$  hinzu.

Nun wird der DYCOS-Algorithmus auf diesen Knoten angewendet und  $v_2$  mit  $A$  beschriftet.

Zum Zeitpunkt  $t = 3$  komme ein Knoten  $v_3$ , der mit  $B$  beschriftet ist, und die Kante  $(v_2, v_3)$  hinzu.

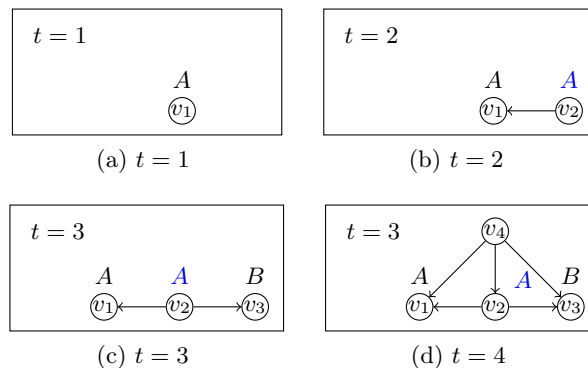


Abbildung 2: Minimalbeispiel für den Einfluss früherer DYCOS-Anwendungen

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  $B$  beschriftet wird. Aber in diesem Beispiel wurde der Knoten schon zum Zeitpunkt  $t = 2$  beschriftet. Obwohl es in diesem kleinen Beispiel noch keine Rolle spielt, wird das Problem klar, wenn man weitere Knoten einfügt:

Wird zum Zeitpunkt  $t = 4$  ein unbeschrifteter 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  $A$  beschriftet wird bei  $\frac{2}{3}$ . Werden die unbeschrifteten Knoten jedoch erst jetzt und alle gemeinsam beschriftet, so ist die Wahrscheinlichkeit für  $A$  als Beschriftung bei nur 50%. Bei dem DYCOS-Algorithmus findet also eine Überanpassung an vergangene Beschriftungen statt.

Das Reklassifizieren von Knoten könnte eine mögliche Lösung für dieses Problem sein. Knoten, die durch den DYCOS-Algorithmus beschriftet wurden könnten eine Lebenszeit bekommen (TTL, Time to Live). Ist diese abgelaufen, wird der DYCOS-Algorithmus erneut auf den Knoten angewendet.

## 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 wodurch mehr Artikel mit einander durch Vokabular verbunden werden können. Außerdem könnte so der Gini-Koeffizient ein besseres Maß für die Gleichheit von Texten werden.

Eine weitere Verbesserungsmöglichkeit besteht in der Textanalyse. Momentan ist diese noch sehr einfach gestrickt und ignoriert die Reihenfolge von Wörtern 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 Beschriftungen.

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.