Knowledge-based WSD – nach Dongsuk et al.

Christian Wurm
Universität Düsseldorf

Überblick

Graphentheorie

Die Kontexte bestimmen

Die Einbettung

Zusammenfassung

Zusammenfassung von

Dongsuk etal.:
Word Sense Disambiguation Based on Word Similarity Calculation Using Word Vector Representation from a Knowledge-based Graph.
2018

Überblick

Die Methode lässt sich in 3 Teile aufspalten:

1. Ambige Worte werden anhand ihrer Kontextworte disambiguiert, mittels eines Graphenalgorithmus, der berechnet:

   welche Reihe von Lesarten Ziel+Kontext hängt am engsten Zusammen im Wissensgraph?

2. Da diese Berechnung exponentiell ist in der Länge der Eingabe (wie fast alle Graphenalgorithmen), ist es wichtig, den Kontext richtig zu wählen (möglichst klein, möglichst relevant)
3. Hierfür nutzen wir word-embeddings, die wiederum auf der Wissensbasis basieren.

Wir betrachten die drei Teile separat.

Der Graph

Graphenstruktur

Ein Graph ist eine Struktur (E, V), mit $E \subseteq V^2$. Wir bekommen aus BabelNet einen Graphen, indem wir nur Knoten (mit identifier) und Kanten behalten, und alle andere Information weglassen.

Sei l eine Funktion, die für jedes Wort die Lesarten liefert:

$$l(w) = \{v : w \in synset(v)\} \tag{1}$$

Wir erinnern uns: gegeben eine Menge von Worten

$$\{w_1, w_2, ..., w_n\}$$

bekommen wir eine **Menge von Mengen**, die Menge der Lesarten:

$$l(\{w_1, ..., w_n\}) = \{\{v_1, ..., v_n\} : v_1 \in l(w_1), ..., v_n \in l(w_n)\} \tag{2}$$

Graphenstruktur

$$l(\{w_1, ..., w_n\}) = \{\{v_1, ..., v_n\} : v_1 \in l(w_1), ..., v_n \in l(w_n)\} \quad (3)$$

Man beachte: $|l(\{w_1, ..., w_n\})|$ wächst **exponentiell** in n!

Hier liegt also das erste computationelle Problem!

Der Graphenalgorithmus

Das Problem

Die Frage ist: welche Menge $\{v_1, ..., v_n\} \in I(\{w_1, ..., w_n\})$ ist am **plausibelsten**?

Diese Frage wird hier graphentheoretisch beantwortet.

Die Lösung des Problems besteht wiederum aus zwei Teilen:

1. Die Konstruktion eines Teilgraphen mit den Knoten $v_1, ..., v_n$.
2. Gegeben eine Menge von Teilgraphen $G_1, ..., G_m$, welcher Teilgraph ist der plausibelste?

Konstruktion der Teilgraphen

Gegeben ein Graph $G = (E, V)$, und eine Menge $X = \{v_1, ..., v_n\} \subseteq V$, konstruiere

$$SG(G, X) = (E', V') \tag{4}$$

wobei

1. $E' \subseteq E$
2. $X \subseteq V' \subseteq V$
3. (E', V') ist ein zusammenhängender Graph
4. (E', V') sollte der minimale Graph sein, der diese Bedingungen erfüllt.

Konstruktion der Teilgraphen

1. $E' \subseteq E$
2. $X \subseteq V' \subseteq V$
3. (E', V') ist ein zusammenhängender Graph
4. (E', V') sollte der minimale Graph sein, der diese Bedingungen erfüllt.

1-3 ist klar: wir möchten einen Teilgraphen konstruieren, der alle Lesarten verbindet.

4. dagegen ist problematisch, und der Text sagt nichts dazu.

Es wird verwiesen auf Manion et al. 2014...

Welcher Teilgraph ist am Plausibelsten?

Nehmen wir also an, $SG : (G, X) \mapsto G'$ ist eine wohldefinierte Funktion.

Daraus folgt: für jedes $X \in I(\{w_1, ..., w_n\}$ bekommen wir einen Teilgraphen. Also:

$$G_1, ..., G_m, \text{ wobei } m \approx c^n,$$

c die durchschnittliche lexikalische Ambiguität.

Wir suchen nun den *besten Graphen*, und die Knoten liefern uns dann jeweils die **besten Lesarten**.

Konnektivität

Sei $G = (E, V)$ ein Graph. Wir können Kanten subtrahieren:

$$G - E' = (E - E', V) \tag{5}$$

Wir sagen, ein Graph unverbunden, wenn er aus zwei separaten Teilen besteht:

$$E \subseteq (V_1)^2 \cup (V_2)^2, \text{ wobei } V_1 \cap V_2 = \emptyset \text{ (und } V_1 \neq \emptyset \neq V_2)$$

Und wir definieren:

$$conn(E, V) = min\{|X| : (E, V) - X \text{ ist unverbunden}\} \tag{6}$$

Algorithmus zur Berechnung

Die Berechnung von $conn(E, V)$ geht wie folgt:

- sei $p(u, v)$ die Anzahl der Pfade von u nach v, die *keinen gemeinsamen Knoten* haben (hier gilt $u \neq v$)
- Dann gilt:

$$conn(E, V) = min_{u,v \in V} p(u, v) \tag{7}$$

- $p(u, v)$ lässt sich schnell berechnen (linear über $|V|$)
- Also: $|V|^2 \cdot |V|$ ($|V|^2$ sind die Paare)

Und hier also die Antwort

Gegeben $G_1, ..., G_m$ gilt:

$$\hat{G} = \underset{G \in \{G_1, ..., G_m\}}{argmax} \; conn(G) \tag{8}$$

Der Zielgraph ist also derjenige mit der höchsten Konnektivität!

Und der Graph, mit seinen Knoten, liefert uns wiederum die Lesarten der Worte.

Kontexte bestimmen

Das Problem

Wir sehen, in der Berechnung von $\hat{G}$ gibt es verschiedene teure Schritte

- Die Anzahl der möglichen Teilgraphen
- Die Berechnung der Konnektivität ist auch teuer, je nach Größe des Teilgraphen.

Also gilt es: einen möglichst kleinen, möglichst relevanten Kontext zu finden.

Das Problem

Die Aufgabe stellt sich wie folgt:

- Gegeben ein Vorkommen von w,
- finde $w_1, ..., w_n$, aus dem Dokument, die die Lesart von w möglichst eindeutig bestimmen.

Wir denotieren diese Menge mit

$$con_n(w) \tag{9}$$

der relevanteste Kontext mit Größe n.

Die Lösung

Wir nehmen an, wir haben für jedes Wort einen zugehörigen Vektor

$$vec(w) \in \mathbb{R}^k \tag{10}$$

also ein klassisches word-embedding.

Wir wählen den relevanten Kontext nach folgender Logik:

Die Logik

Je semantisch ähnlicher ein Wort dem Zielwort ist, desto eher ist es relevant.

Die Lösung

Die Logik

Je semantisch ähnlicher ein Wort dem Zielwort ist, desto eher ist es relevant.

Semantische ist hier aber ein sehr gut bekanntes Konzept, man operationalisiert es mit dem **Kosinus** zweier Vektoren:

$$cos(\vec{x}, \vec{y}) = 1 \Longrightarrow \text{ selbe Richtung} \tag{11}$$

$$cos(\vec{x}, \vec{y}) = 0 \Longrightarrow \text{ orthogonal} \tag{12}$$

$cos(\angle) = (Ankathete/Hypothenuse)$

Die Lösung: Kosinus

Der Kosinus von $\vec{x}, \vec{y}$ berechnet sich als

$$cos(\vec{x}, \vec{y}) = \frac{\vec{x}^\top \vec{y}}{\|\vec{x}\| \cdot \|\vec{y}\|} \tag{13}$$

Also bekommen wir:

$$con_n(w) = \{w' : cos(vec(w), vec(w')) \text{ ist in den maximalen } n\text{, f.a. } w' \text{ im Dokument}\}$$

Die Einbettung

Die Einbettung

Bleibt die Frage nach der Einbettung. Das word-embedding werden wir nicht besprechen, hier gibt es viele fertige Algorithmen.

Aber woher kommen die Daten fürs embedding?

Die Einbettung

Random Walk

Man generiert sich das Korpus mittels *random walk* über BabelNet, wobei man

- An einem zufällig gewählten Knoten anfängt,
- entlang einer Kante zu einem zufällig gewählten Knoten weiterwandert
- und dabei jedesmal zufällig ein Wort aus dem entsprechenden Synset auswählt
- oder alternativ zum weitergehen den so generierten "Satz" beendet.

Die Einbettung

Auf diese Weise kann man ein semantische informatives Korpus aus Babelnet generieren, auf dem das word-embedding trainiert wird!

Das dauert natürlich seine Weile, ist aber ein *offline-task*, der nur einmal gemacht werden muss.

Zusammenfassung

Der Ablauf

- (offline) Wir erzeugen das Korpus aus BabelNet
- (offline) Wir trainieren das embedding auf dem Korpus
- Gegeben ein Dokument, und ein Zielwort, wählen wir den Kontext mit *cos* auf den embeddings
- Gegeben Zielwort und Kontext, erzeugen wir Teilgraphen in BabelNet
- Mittels *connectivity* suchen wir den plausibelsten Teilgraphen $\hat{G}$
- Mittel $\hat{G}$ bekommen wir die Lesarten des Zielwortes und der Kontextworte!