Patent – Triz40 Mappingansätze

René Brückner

30. September 2019

Zusammenfassung

Im Laufe des Seminars *Widersprüche und Management-Methodiken* an der Uni Leipzig im Sommersemester 2019 wurden die Studierenden immer wieder vor die Problematik der Zuordnung von Patenttexten zu Innovationsmethoden der 40 TRIZ Prinzipien gestellt. Als Beitrag des Forschungsseminars hat sich der Autor das Ziel gestellt, mittels Textprocessing und Information Retrieval Methoden zu Patenttexten eine Auswahl möglicher der 40 TRIZ Methoden automatisiert zu ermitteln. Zu Beginn werden Grundlagen abgesteckt, welche für das weitere Verständnis erforderlich sind. Anschließend wird eine Zielstellung formuliert und auf die Ausgangssituation eingegangen. Auf Grundlage dieser beiden Kapitel werden drei Ansätze und deren entsprechenden Einschränkungen sowie Probleme vorgestellt. Aufgrund des Umfangs des Moduls ist eine Evaluierung der Ansätze nicht durchführbar. Zum Schluss wird ein Ausblick für eine mögliche weitere Bearbeitung des Themas gegeben.

Universität Leipzig
Sommersemester 2019

Fakultät für Mathematik und Informatik
Seminar Widersprüche und Management-Methodiken

Inhaltsverzeichnis

1 Einleitung

Mithilfe von Ansätzen, die Altshuller in seinem Buch [2] beschreibt, lassen sich Konflikte im Innovationsprozess überwinden. Seine 40 Methoden fanden Anwendung in vielen modernen Technologien und Erfindungen. Spuren dieses Einflusses lassen sich erkennen, indem man Patenttexte auf Parallelen zu seinen Prinzipien hin untersucht. Dies war Hauptbestantteil des Seminars *Widersprüche und Management-Methodiken* der Uni Leipzig im Sommersemester 2019. Die Recherche entsprechender Patente stellte dabei eine nicht triviale Aufgabe für die Teilnehmer wie den Autor. Eine automatisierte Zuordnung erleichtert die Forschung und die Analyse bestehender Konfliktmanagementmethoden. Moderne Text Mining und Information Retrieval Methoden haben eine Vielzahl von Algorithmen zum Textvergleich als Grundlage. Der Autor hat sich das Ziel gestellt, solche Algorithmen anzuwenden, um zu erproben, wie eine automatische Zuordnung der 40 TRIZ Prinzipien zu Patenttexten umgesetzt werden könnte. Dazu stellt der Autor zunächst kurz verwendete Grundlagen vor, um im folgenden drei Ansätze zu erläutern.

2 Grundlagen

Zum Verständnis der Ansätze sind Grundlagen im Bereich des Text Minings, Information Retrieval sowie mathematische Ähnlichkeitsmaße notwendig. Im Folgenden werden diese kurz benannt und definiert jedoch nicht im Detail erläutert. Für genaue Ausführungen verweist der Autor auf die an der Leipziger Universität gehaltenen Vorlesungen *Text Mining* und *Information Retrieval* sowie die Bücher [6] und [3].

2.1 Text Mining / Information Retrieval Begriffserklärung

- **Tokenizing:** Extrahieren einzelner Wörter (tokens) eines Satzes
- **Stopwort-Entfernung:** Entfernen semantisch unwichtiger Wörter (z.B. *und, der, die, das, als,* etc.) via Listen.
- **Stemming:** Meist regelbasierte Reduzierung des Worts auf den Wortstamm (*Häuser Haus*, *gesprungen springen* ...).
- **POS Tagging:** *Part Of Speech Tagging* ist die Bestimmung der Wortart im Satz.

2.2 Mathematische Grundlagen

Eine naive Form der Ähnlichkeitdefinition ist der *Jaccard-Index* [5]

$$J_\delta(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}.$$

Dieser liegt zwischen 0 und 1 und ist ein Maß für die Ähnlichkeit der beiden (endlichen) Mengen A und B.

Die Kosinusähnlichkeit [9] ist ein besseres Maß zur Ähnlichkeitsbestimmung und fundiert auf der linearen Algebra. Dabei wird der Kosinus des Winkels zwischen zwei Vektoren bestimmt, die das Dokument repräsentieren. Mathematisch ausgedrückt entspricht die Kosinusähnlichkeit der folgenden Formel.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||} = \frac{\sum_i A_i B_i}{\sqrt{\sum_i A_i^2}\sqrt{\sum_i B_i^2}}.$$

3 Zielstellung

Grundlegend ist das Ziel, semantische Ähnlichkeiten zwischen einem Patenttext und allen 40 TRIZ Methoden zu quantifizieren. Im Fokus stand für den Autor eher die Erprobung verschiedener Verfahren als eine genaue Einordnung. Die Zuordnung zu möglichen Kandidaten ist dabei wichtiger als eine exakte Bestimmung aller angewandten Methoden. Zunächst wurde die Beschaffenheit der verschiedenen Texte betrachtet, um eine Abschätzung der Umsetzbarkeit zu ermöglichen. Patenttexte sind in einem sehr speziellen Vokabular verfasst. Beschreibende technische Bezeichnungen und Erläuterungen ihrer Interaktionen untereinander sowie mit einem potenziellen Nutzer stehen im Mittelpunkt. Aufgrund dieser Besonderheiten hat jeder Text markante Eigenschaften, welche einen Vergleich ermöglichen.

Die 40 TRIZ Methoden hingegen haben eine allgemein gehaltene Beschreibung ohne Konkretisierung von entsprechenden Bauteilen. Verwendet wurde eine detaillierte Übersetzung von Altschullers Prinzipien von *www.triz40.com* [4] ins Englische sowie ins Deutsche. Aufgrund dieser allgemein gültigen Formulierungen besteht die Möglichkeit, Parallelen zwischen der entsprechenden Methode und einem Patenttext zu ziehen.

Ziel ist nun, ein Verfahren zu finden, welches möglichst viele treffende Kandidaten der 40 TRIZ Methoden einem Patent zuordnen kann. Aufgrund von fehlenden Daten zum Vergleich und mangelnder Bearbeitungszeit ist es dem Autor nicht möglich, eine genaue Evaluierung der Ergebnisse vorzunehmen. Vielmehr sollen die Ansätze einen Einstieg für eine weitere Bearbeitung dieses Themas bieten.

4 Ansätze

Für die Präsentation während des Seminars sowie zur Erprobung wurden beispielhaft drei Ansätze in Python implementiert. Zur Erleichterung wurde ein Modul erstellt, welches die 40 TRIZ Methoden in Englisch sowie Deutsch enthält. Diese wurden Patenttexten gegenübergestellt. Inspiration der Verwendung dieser Textvergleiche und zugleich Grundlage bildeten die Artikel [8] und [1]. Die erstellten Skripte sind bei Github[1] zu finden. In den folgenden Kapiteln werden sie genauer erläutert.

[1] Im Verzeichnis `Sommersemester-2019/2019-07-04/Mappingansaetze/` des github Repos `https://github.com/wumm-project/Leipzig-Seminar`.

4.1 TF-IDF

Bei der Termfrequenz – Inverse Dokumenten Frequenz (TF-IDF) wird das Auftreten einzelner Terme eines Dokuments in einer Matrix abgebildet. Genauer wird das Vorgehen in [3] erläutert. Es handelt sich um eine Methode des Text Mining und des Information Retrieval, welche ursprünglich für Suchmachinen entwickelt wurde.

Zur Umsetzung wurde die Python Implementierung von *sklearn* verwendet. Diese unterstützt das Entfernen von Stopwörtern, welche das Ergebnis verfälschen. Beispielsweise würden Texte, welche viele Stopwörter verwenden, sonst automatisch als ähnlicher eingeschätzt. Die Implementierung erfolgte wie in folgendem Ausschnitt.

```
1 def tfidf(text, compareset, stopwords):
2     vect = TfidfVectorizer(min_df = 1, stop_words = stopwords)
3     tfidf = vect.fit_transform([text] + compareset)
4     return((tfidf * tfidf.T).A[0][1:])
```

Die Funktion gibt ein Array mit 40 Zahlenwerten zwischen 0 und 1 zurück, wobei Eins die größte und Null die kleinste Ähnlichkeit ausdrückt. In Zeile 2 wird das minimale Vorkommen der zu berücksichtigenden Wörter festgelegt und zu entfernende Stopwörter übergeben. In Zeile 3 wird der Vergleich ausgeführt und in einer Matrix abgebildet, wovon schließlich in Zeile 4 die erste Reihe ausgegeben wird.

Mehrdeutigkeiten oder andere Formulierungen können das Ergebnis stark verfälschen, da diese nicht berücksichtigt werden.

4.2 Wordnet

Ein etwas komplexerer Ansatz ist die Realisierung eines Vergleichs beider Texte unter der Berücksichtigung von möglicher Mehrdeutigkeit und semantischer Ähnlichkeit der Wörter. Wordnet ist ein lexikalisch semantisches Netzwerk, welches 1998 von George Miller in [7] vorgestellt wurde. Es ermöglicht, die Bedeutungsähnlichkeit zwischen Wörtern zu quantifizieren. Da unsere Zielstellung jedoch Textvergleiche und nicht Wortvergleiche voraussetzt, bedient sich der Autor einer in [6] vorgestellten Formel für Textvergleiche:

$$\mathrm{Sim}(T_1, T_2) = \frac{1}{2}\left(\frac{\sum_{w \in T_1} \mathrm{maxSim}(w, T_2) \cdot idf(w)}{\sum_{w \in T_1} idf(w)} + \frac{\sum_{w \in T_2} \mathrm{maxSim}(w, T_1) \cdot idf(w)}{\sum_{w \in T_2} idf(w)}\right).$$

Dies ermöglicht Texte über ihre eigentliche Formulierung hinaus semantisch gegenüberzustellen. Dieses Vorgehen wurde mithilfe der Python Bibliothek *nltk* umgesetzt, welche Wordnet beinhaltet wie im folgenden Quelltextabschnitt beschrieben.

```
1 #sentence similarity using Wordnet
2 def sentence_similarity(sentence1, sentence2):
3     # Tokenize and tag
```

```
4       sentence1 = pos_tag(word_tokenize(sentence1))
5       sentence2 = pos_tag(word_tokenize(sentence2))
6       # Get the synsets for the tagged words
7       synsets1 = [tagged_to_synset(*tagged_word) for tagged_word in sentence1]
8       synsets2 = [tagged_to_synset(*tagged_word) for tagged_word in sentence2]
9       # Filter out the Nones
10      synsets1 = [ss for ss in synsets1 if ss]
11      synsets2 = [ss for ss in synsets2 if ss]
12      score, count = 0.0, 0
13      # For each word in the first sentence
14      for synset in synsets1:
15          # Get the similarity value of the most similar word in the other
            ↪ sentence
16          scores = []
17          for ss in synsets2:
18              simscore = synset.path_similarity(ss)
19              if simscore == None:
20                  scores.append(0)
21              else:
22                  scores.append(simscore)
23          best_score = max(scores)
24          # Check that the similarity could have been computed
25          if best_score is not None:
26              score += best_score
27              count += 1
28      # Average the values
29      score /= count
30      return score
31
32  def symmetric_sentence_similarity(sentence1, sentence2):
33      return (sentence_similarity(sentence1, sentence2) +
        ↪ sentence_similarity(sentence2, sentence1)) / 2
```

In Zeile 4 bis 12 werden die Wörter der Dokumente extrahiert, mit ihrem POS-Tag versehen und entsprechende Synonymmengen erstellt. In der Schleife ab Zeile 14 wird die Berechnung des einen Summanden der Formel umgesetzt. Schließlich nutzt die Funktion in Zeile 32 die Summanden, um die Berechnung durchzuführen.

Problematisch bei diesem Ansatz ist, dass der Vergleich der Synonyme abhängig von dem von Wordnet bereitgestellten semantischen Netz ist. Eine technische Beschreibung von Patenten hat ein sehr spezielles Vokabular, dessen Synonyme nicht im Netz enthalten sein könnten.

5 Word2Vec

Als letzter Ansatz wurde ein auf Machine Learning basierender Ansatz verfolgt. Dazu wird zunächst ein Beispielmodell trainiert. Als Trainingsdaten wurden in diesem Fall ausschließlich

die Texte der 40 TRIZ Methoden verwendet. Empfehlenswert wäre ein Datensatz angemessener Größe, um die komplexen Zusammenhänge der technischen Beschreibungen und Methoden zu erfassen.

Die Funktionalitäten der Modellerstellung sowie der eigentliche Vergleich, mittels Kosinusähnlichkeit wurden mit der Python Bibliothek *gensim* realisiert.

```
def trainModel():
    sentences = []
    for methode in methoden:
        sentences.append(methode.split(' '))
    model = gensim.models.Word2Vec(sentences, min_count=1)
    model.wv.save('./models/model.bin')

def w2vCompare(s1, s2, wordmodel):
    s1wordsset = set(s1.split())
    s2wordsset = set(s2.split())
    for word in s1wordsset.copy():
        if (word not in wordmodel.vocab):
            try:
                s1wordsset.remove(word)
            except KeyError:
                pass
    for word in s2wordsset.copy():
        if (word not in wordmodel.vocab):
            try:
                s2wordsset.remove(word)
            except KeyError:
                pass
    return wordmodel.n_similarity(list(s1wordsset), list(s2wordsset))
```

Die Funktion *trainModel()* benuzt die Stringoperation Split, um die Texte in Token umzuwandeln. Diese werden genutzt, um ein Modell zu trainieren und selbiges zu speichern. Die Funktion *w2vCompare* bekommt als Parameter zwei zu vergleichende Texte und ein Modell, um eine Schnittmenge aus den Vokabularen als Vektorraum aufzuspannen und mittels Kosinusähnlichkeit zu vergleichen.

Der Vorteil dieses Ansatzes ist, dass er die Schwächen der beiden letzten Vorgehen berücksichtigt. Jedoch geschieht dies aufgrund der Abhängigkeit von einem zu trainierenden Modell. Wie bereits zuvor erwähnt sollte die Modellerstellung wesentlich mehr Texte beinhalten.

6 Ausblick

Die vorgestellten Ansätze sind nicht in der Lage, ein komplexes Mapping gut zu realisieren. Jedoch bieten sie einen Einblick in die Möglichkeiten einer Umsetzung. Im Grunde ist die

vorgestellte Problemstellung der Klassifizierung sehr gut geeignet für ein neuronales Netz. Neuronale Netze sind state-of-the-art Methoden zum Lösen von Klassifizierungsproblemen. Dazu könnte auf dem in 4.3 vorgestellten Ansatz aufgebaut werden. Für das Seminar war dies jedoch nicht umsetzbar, da die Qualität der Klassifizierung mittels eines neuronalen Netzes immer von der Menge der Trainingsdaten abhängt. Um einen großen Trainingsdatensatz zu erstellen, müsste eine händische Klassifizierung vorgenommen werden. Alternativ könnte man einen der anderen Ansätze verwenden, um eine Vorauswahl zu treffen, welche dann evaluiert in den Trainingsdatensatz mit einbezogen wird.

Anwendungsmöglichkeiten für diese Verfahren wäre die Patentrecherche sowie weitere Forschung bezüglich angewandter Innovationsmethoden. Auch eine Untersuchung, ob ein Patentkandidat notwendige Innovation aufweist, wäre vorstellbar. Zuletzt könnte ein bestehendes neuronales Netz um Konfliktarten erweitert werden, um entsprechende Innovationsmethoden zur Konfliktlösung vorzuschlagen.

Insgesamt ist hervorzuheben, dass dieser Bereich der Innovationsforschung im Bezug auf Methoden der modernen Informatik, speziell bezüglich Big Data und Machine Learning, noch sehr viel Potenzial hat.

Literatur

[1] Sieg Adrien. Text similarities : Estimate the degree of similarity between two texts, 2018.

[2] Genrich Altshuller. *40 principles: TRIZ keys to innovation*, volume 1. Technical Innovation Center, Inc., 2002.

[3] Ricardo Baeza-Yates, Berthier Ribeiro-Neto et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[4] Olivier Goguel. triz40.com, 2014.

[5] Paul Jaccard. *Lois de distribution florale dans la zone alpine*. 1902.

[6] Rada Mihalcea, Courtney Corley, Carlo Strapparava et al. Corpus-based and knowledge-based measures of text semantic similarity. In *Aaai*, volume 6, pages 775–780, 2006.

[7] George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

[8] Maali Mnasri. Quick review on text clustering and text similarity approaches, 2016.

[9] Grigori Sidorov, Alexander Gelbukh, Helena Gómez-Adorno, David Pinto. Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas*, 18(3):491–504, 2014.