Übung: Regelbasierte Textanalyse

Am Beispiel der Tagesschau-Daten

Prof. Dr. Nicolas Meseth

Text in Nachrichtenartikeln ist unstrukturiert, aber nicht formlos. Überschriften folgen Mustern. Autoren wiederholen sich. Themen tauchen auf, verschwinden und kehren wieder. Regelbasierte Textanalyse macht genau diese Muster sichtbar, indem sie auf rohen Text Struktur projiziert: durch Suchen, Extrahieren, Transformieren und Zählen. In dieser Übung baut ihr mit dem Tagesschau-Datensatz eine vollständige Analysepipeline auf, vom einfachen Keyword-Suchen bis zur dictionary-basierten Themenklassifikation über zwei Jahrzehnte Nachrichtengeschichte.

Schritt 1: Pakete laden und Daten einlesen

1. Installiert und ladet mit `pacman::p_load()` die folgenden Pakete: `tidyverse`, `skimr`, `tidytext`, `jsonlite`, `stopwords`. Beschreibt in einem Satz, wozu `tidytext`, `jsonlite` und `stopwords` jeweils eingesetzt werden.

2. Lest den Tagesschau-Datensatz mit `read_csv()` ein. Ergänzt direkt im selben Pipeline-Schritt zwei neue Variablen: `year` (Erscheinungsjahr aus `date_time` mit `lubridate::year()`) und `date` (nur das Datum als `as.Date(date_time)`).

Welche Spalten des Datensatzes enthalten freien Text, welche strukturierte Werte?

Schritt 2: Texte durchsuchen und klassifizieren

3. Nutzt `str_detect()`, um vier neue logische Spalten zu erzeugen:

- `thema_klima` (Muster `"Klima"`),
- `thema_corona` (Muster `"Corona|COVID"`),
- `thema_ukraine` (Muster `"Ukraine"`) und
- `thema_ki` (Muster `"\bKI\b|[Kk]ünstliche Intelligenz"`).

Sucht jeweils in der `title`-Spalte. Wie viele Artikel enthalten pro Thema das Muster?

4. Stellt die zeitliche Entwicklung aller vier Themen in einem Liniendiagramm dar. Berechnet zunächst die Anzahl der Treffer pro Jahr und Thema, überführt das Ergebnis mit `pivot_longer()` in Langform und plottet es mit `ggplot2`. Wann war Corona am stärksten präsent? Wann beginnt das Thema KI merklich zu wachsen?

5. Die absoluten Zählungen aus Aufgabe 4 können irreführend sein: In späteren Jahren erscheinen generell mehr Artikel, sodass ein absoluter Anstieg auch einfach das gestiegene Gesamtvolumen widerspiegeln kann, statt eines echten inhaltlichen Bedeutungsgewinns. Berechnet daher die *relative Häufigkeit* jedes Themas pro Jahr als Anteil aller Artikel des jeweiligen Jahres in Prozent. Stellt die relativen Frequenzen als Liniendiagramm dar.

Vergleicht das Ergebnis mit dem Diagramm aus Aufgabe 4. Bei welchen Themen verändert sich das Bild? Was lässt sich nur aus der relativen Darstellung schließen?

6. `stringr` bietet neben `str_detect()` auch `str_starts()` und `str_ends()`. Nutzt `str_starts()`, um Artikel zu finden, deren Titel mit `"Interview:"` oder mit `"Liveblog:"` beginnen. Nutzt `str_ends()`, um Titel zu finden, die mit einem Fragezeichen enden. Wie viele Treffer liefern die drei Muster jeweils? Was ist der Unterschied zu `str_detect()` für dasselbe Suchmuster?

7. Viele Tagesschau-Titel folgen dem Muster `"Format: Überschrift"`. Erstellt eine neue Variable `format`, die den Artikeltyp klassifiziert. Nutzt `case_when()` in Kombination mit `str_starts()` und Vergleichen auf die `tag`-Spalte. Unterscheidet mindestens: *Interview*, *FAQ*, *Analyse*, *Hintergrund*, *Kommentar*, *Liveblog* und *Meldung* als Auffangkategorie. Wie häufig ist jedes Format?

8. Stellt die Formatverteilung für die vier größten Ressorts (`ausland`, `wirtschaft`, `inland`, `wissen`) als gestapeltes Balkendiagramm mit relativen Anteilen dar. Nutzt dafür `geom_col(position = "fill")` oder berechnet die Anteile explizit mit `mutate(anteil = n / sum(n))`. Welches Ressort hat den höchsten Anteil an FAQ-Artikeln? Welches an Analysen?

Schritt 3: Texte bereinigen und strukturieren

Reale Datensätze sind selten sauber. Die Tagesschau-Daten enthalten HTML-Fragmente in Autorennamen, fehlerhafte Ressort-Einträge und uneinheitliche Schreibweisen. Bevor ihr

analysiert, müsst ihr bereinigen.

9. Schaut euch die `ressort`-Spalte genau an: Gebt alle Ressorts mit weniger als 100 Artikeln aus. Welche Einträge sind offensichtlich fehlerhaft oder technischen Ursprungs? Erstellt anschließend eine bereinigte Version `news_clean`, die nur Artikel mit sinnvollen Ressort-Werten enthält. Wie viele Zeilen werden dabei entfernt?

10. Die Spalte `author` enthält viele Unregelmäßigkeiten: HTML-Tags wie `<em>...</em>`, Präfixe wie `"Von "` und Quellenangaben wie `", ARD-Studio Brüssel"`. Bereinigt die Spalte schrittweise und speichert das Ergebnis als `author_clean`. Wendet folgende Schritte der Reihe nach an: (1) HTML-Tags entfernen, (2) führendes `"Von "` entfernen, (3) alles ab dem ersten Komma entfernen, (4) Leerzeichen normalisieren mit `str_squish()`. Überprüft das Ergebnis mit einigen Beispielzeilen.

11. Ermittelt mit der bereinigten `author_clean`-Spalte die 15 produktivsten Autorinnen und Autoren im Datensatz. Filtert leere Strings und `NA`-Werte heraus. Stellt das Ergebnis als horizontales Balkendiagramm dar, sortiert nach Häufigkeit. Welche Redaktionen oder Studios tauchen auf?

12. Viele Tagesschau-Titel folgen dem Muster `"Stichwort: Eigentliche Überschrift"`. Trennt dieses Muster auf: Erstellt eine Variable `stichwort` (Text vor dem ersten Doppelpunkt, bereinigt mit `str_trim()`) und eine Variable `ueberschrift` (Text nach dem Doppelpunkt). Für Artikel ohne Doppelpunkt soll `stichwort` `NA` sein und `ueberschrift` dem vollen Titel entsprechen. Welche zehn Stichworte kommen am häufigsten vor?

13. Reguläre Ausdrücke (*Regex*) ermöglichen komplexere Suchmuster. Sucht in der `title`-Spalte nach:

- (a) Prozentzahlen wie `"47 Prozent"` oder `"3,5 Prozent"`
- (b) Zitate in Anführungszeichen

Wie viele Treffer gibt es jeweils? Gebt fünf Beispieltitel pro Kategorie aus.

Schritt 4: Strukturierte Textfelder parsen

Manche Spalten sind keine einfachen Zeichenketten, sondern serialisierte Datenstrukturen: JSON-Arrays, die als Text gespeichert wurden. `jsonlite::fromJSON()` macht daraus echte R-Objekte, die ihr dann mit den gewohnten tidyverse-Werkzeugen weiterverarbeiten könnt.

14. Die Spalte `keywords` enthält JSON-Arrays wie `["EU", "Trump", "Ukraine"]`. Parst diese Spalte mit `map()` und `jsonlite::fromJSON()` und bringt den Datensatz mit `unnest()` in eine Langform, sodass jede Zeile ein Keyword enthält. Nutzt `possibly()` aus `purrr`, um fehlerhafte JSON-Strings sicher abzufangen. Wie viele Keyword-Einträge gibt es insgesamt? Erstellt eine Rangliste der 20 häufigsten Keywords als Balkendiagramm.

15. Wählt vier Keywords aus (z.B. `"Trump"`, `"Coronavirus"`, `"Ukraine"`, `"Klimawandel"`) und stellt deren Häufigkeit pro Jahr als Liniendiagramm dar. Filtert auf Artikel ab 2010. Lässt sich der Beginn des Ukraine-Kriegs 2022 ablesen? Wann verschwand Coronavirus von der Agenda?

16. Die Spalte `related_links` enthält JSON-Arrays von Objekten. Parst diese Spalte analog zu `keywords`, sodass ihr eine Zeile pro Link habt. Extrahiert aus den Link-URLs mit einem regulären Ausdruck die Domain und erstellt eine Rangliste der 15 häufigsten externen Domains (ohne `tagesschau.de` und `ard.de`).

Schritt 5: Tokenisierung und Worthäufigkeiten

Für eine systematische Textanalyse zerlegt man Texte in einzelne *Tokens*, die kleinsten bedeutungstragenden Einheiten. Das `tidytext`-Paket fügt sich dabei nahtlos in den Tidyverse-Workflow ein und macht Texte tabellenartig behandelbar.

17. Tokenisiert die `title`-Spalte mit `unnest_tokens()` aus `tidytext`. Behaltet dabei `url`, `year` und `ressort`. Was macht `unnest_tokens()` automatisch mit Groß- und Kleinschreibung und mit Satzzeichen? Wie viele Tokens enthält der gesamte Datensatz?

18. Ladet mit eine deutsche Stoppwortliste und entfernt diese aus den Daten mit `anti_join()`. Filtert zusätzlich alle Tokens heraus, die kürzer als drei Zeichen sind oder ausschließlich aus Ziffern bestehen (`str_detect(word, "^\\d+$")`). Wie viele Tokens bleiben nach der Bereinigung übrig?

19. Ermittelt die 20 häufigsten Wörter in den Tagesschau-Überschriften nach der Bereinigung und stellt sie als horizontales Balkendiagramm dar. Welche Wörter erscheinen, und entspricht das euren Erwartungen? Gibt es Wörter in der Liste, die ihr für inhaltlich wenig aussagekräftig haltet und die noch herausgefiltert werden könnten?

20. Vergleicht die häufigsten Wörter in den vier größten Ressorts (`ausland`, `wirtschaft`, `inland`, `wissen`) in einem facettierten Diagramm mit je zehn Wörtern pro Ressort. Nutzt `reorder_within()` und `scale_x_reordered()` aus `tidytext`, damit die Wörter innerhalb jedes Facetts korrekt sortiert sind. Was verraten die Unterschiede über die inhaltlichen Schwerpunkte der Ressorts?

21. Die reine Häufigkeit bevorzugt Wörter, die in allen Ressorts vorkommen. *TF-IDF* (Term Frequency – Inverse Document Frequency) gewichtet Wörter so, dass charakteristische, ressortspezifische Begriffe hervorgehoben werden. Berechnet TF-IDF mit `bind_tf_idf(word, ressort, n)` aus `tidytext` und stellt die jeweils fünf charakteristischsten Wörter pro Ressort als facettiertes Balkendiagramm dar.

22. Vergleicht die TF-IDF-Ergebnisse mit den reinen Häufigkeiten aus Aufgabe 19 für ein oder zwei Ressorts. Welche Wörter sind in beiden Listen? Welche tauchen nur bei TF-IDF auf? Was sagt euch das über die Stärken und Schwächen beider Maße als Beschreibung von Textinhalten?

Schritt 6: Wörterbuch-basierte Klassifikation und Reflexion

Bisher habt ihr einzelne Muster untersucht. Jetzt geht es um systematische Klassifikation: Ihr erstellt ein Themen-Dictionary und wendet es auf den gesamten Datensatz an. Das ist *deduktive Inhaltsanalyse* – ihr bringt eure theoretischen Kategorien mit und sucht sie im Text.

23. Erstellt ein Themen-Dictionary in Excel und ladet es als Tibble mit den Spalten `word` (Kleinbuchstaben) und `thema`. Deckt mindestens vier Themen ab: *Klimapolitik*, *Sicherheit & Krieg*, *Wirtschaft & Finanzen* und *Gesundheit*.

Verwendet pro Thema mindestens fünf charakteristische Wörter. Welche Wörter könnten zu Fehlzuordnungen führen?

24. Wendet das Dictionary auf die tokenisierten Titel an. Nutzt `left_join()` zwischen dem bereinigten Token-Datensatz und dem Dictionary. Wie viele Artikel lassen sich den vier Themen zuordnen?

25. Stellt die Entwicklung aller vier Themen ab 2006 als Liniendiagramm dar. Beschreibt Auffälligkeiten in der zeitlichen Entwicklung der Themen.

26. Ruft zehn zufällige Artikel auf, die dem Thema *Klimapolitik* zugeordnet wurden (`slice_sample(n = 10)`), und inspiziert ihre Titel. Sind die Zuordnungen korrekt? Gebt je ein konkretes Beispiel für einen möglichen *falsch-positiven* Treffer (Artikel wird fälschlich zugeordnet) und einen möglichen *falsch-negativen* Treffer (Artikel müsste zugeordnet werden, wird es aber nicht).

Was könnt ihr tun, um das Ergebnis zu verbessern?

27. Die regelbasierte Textanalyse, die ihr in dieser Übung angewendet habt, ist eine Form der *deduktiven Inhaltsanalyse*. Diskutiert die folgenden Fragen:

- Was sind die zentralen Stärken dieses Ansatzes gegenüber manuellem Durchlesen?
- Was wäre ein *induktiver* Ansatz (z.B. Topic Modeling), und in welcher Situation würdet ihr ihn bevorzugen?
- Nennt zwei Situationen, in denen regelbasierte Textanalyse klar an ihre Grenzen stößt.