HAW HAMBURG

BACHELOR THESIS
Kristoffer Schaaf

Entwicklung einer Software zur Erkennung von Fake News auf Nachrichtenportalen

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Kristoffer Schaaf

Entwicklung einer Software zur Erkennung von Fake News auf Nachrichtenportalen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Angewandte Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Stefan Sarstedt
Zweitgutachter: Prof. Dr. Marina Tropmann-Frick

Eingereicht am: 01.04.2025

Kristoffer Schaaf

Thema der Arbeit

Entwicklung einer Software zur Erkennung von Fake News auf Nachrichtenportalen

Stichworte

Machinelles Lernen, Fake News, Nachrichtenportale,

Kurzzusammenfassung

Arthur Dents Reise in eine neue Zukunft ...

Kristoffer Schaaf

Title of Thesis

Development of a software for the detection of fake news on news portals

Keywords

Machine Learning, Fake News, Text Mining, Classification, NLP

Abstract

Arthur Dents travel to a new future ...

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Hintergrund: Die zunehmende Verbreitung von Fake News und deren gesellschaftliche Auswirkungen

1.1.1 Wann entstanden Fake News

Fake News sind ein allgegenwärtiges Problem, doch hatten Sie Ihren ersten Auftritt bereits 44BC im römischen Reich [**?**]. Auch während des amerikanischen Bürgerkriegs 1779 wurden Sie als politischer Schachzug von Benjamin Franklin genutzt. Dieser schickte einen Brief an Captain Samuel Gerrish und schrieb in diesem über Grausamkeiten der Briten und deren Verbündeten. Diese Informationen wurde so veranschaulicht, dass sie die öffentliche Meinung bewusst beeinflussen sollten [**?**].

Der eigentliche Begriff "Fake News"wurde erst viele Jahre später durch Donald Trump im amerikanischen Wahlkampf 2016 bekannt [**?**] und diente hierbei als politischer Kampfbegriff [**?**].

Unter anderem ist Fake News auch ein Teil von Propaganda [**?**], welche schon lange als Mittel zur Meinungsmanipulation eines Volkes genutzt wird.

Heute ist Fake News die größte Drohung zu unserer angeblich freien Presse [**?**].

1.1.2 Wie definieren sich Fake News und wie sind sie aufgebaut

Fake News sind bewusst erstellte Online-Falschmeldungen, die teilweise oder vollständig unwahre Inhalte verbreiten, um Leser*innen gezielt zu täuschen oder zu manipulieren. Sie imitieren klassische Nachrichtenformate, nutzen auffällige Titel, emotionale Bilder und strategisch gestaltete Inhalte, um Glaubwürdigkeit zu erzeugen und Aufmerksamkeit zu

gewinnen. Ziel ist es, durch das Verbreiten dieser Inhalte Klicks, Reichweite und damit finanzielle oder ideologische Vorteile zu erzielen [**?**].

Fake News fallen in die Kategorien Satire, Clickbait, Gerüchte, Stance News, Propaganda und Large Scale Hoaxes [**?**].

- **Satire**: ist eine humorvolle oder übertriebene Darstellung gesellschaftlicher oder politischer Themen, die Kritik üben soll.
- **Clickbait**: bezeichnet reißerische Überschriften oder Vorschaubilder, die Neugier wecken und zum Anklicken eines Inhalts verleiten sollen, oft ohne den Erwartungen gerecht zu werden.
- **Gerüchte**: sind unbestätigte Informationen, die sich schnell verbreiten und oft falsch oder irreführend sind.
- **Stance News**: sind Nachrichten, die eine klare Meinung oder politische Haltung einnehmen, statt neutral zu berichten.
- **Propaganda**: ist die gezielte Verbreitung von Informationen oder Meinungen, um das Denken und Handeln von Menschen zu beeinflussen, meist im Interesse einer bestimmten Gruppe oder Ideologie.
- **Large Scale Hoaxes**: sind absichtlich erfundene Falschmeldungen oder Täuschungen, die weit verbreitet werden und viele Menschen täuschen sollen.

Die eigentliche Nachricht ist aufgebaut in folgende Teile:

- **Quelle**: gibt den Ersteller der Nachricht an.
- **Titel**: erzielt die Aufmerksamkeit der Lesenden.
- **Text**: enthält die eigentliche Information der Nachricht.
- **Medien**: in Form von Bildern oder Videos.

Fake News können die Form von Text, Fotos, Filmen oder Audio annehmen und sind dementsprechend auf jeder Platform auffindbar, die die Verbreitung nicht unterbindet. Die 2024 populärste Platform zum Teilen der Fake News ist WhatsApp [**?**].

1.1.3 Aus welcher Motivation entstehen Fake News

Das Hauptinteresse der Ersteller der Fake News ist das Verdienen von Geld. Auf die Artikel wird Werbung geschaltet und anhand einer entsprechenden Reichweite ergibt sich der verdiente Betrag. Je mehr Reichweite, desto mehr Verdienst für die Ersteller [**?**].

1.1.4 Warum verbreiten sich Fake News

In sozialen Medien neigen Nutzer aufgrund von FOMO (Fear of Missing Out) dazu, Fake News zu teilen, um Anerkennung zu gewinnen und soziale Zugehörigkeit zu erfahren. Besonders häufig werden kontroverse, überraschende oder bizarre Inhalte verbreitet – insbesondere dann, wenn sie starke Emotionen wie Freude, Wut oder Aufregung hervorrufen. Das Teilen solcher Inhalte stärkt das eigene Ansehen, da es signalisiert, über neue und relevante Informationen zu verfügen. Fake News bestehen meist aus eindrucksvoll präsentierten Falschinformationen [**?**].

Ein Grund für die schnelle Verbreitung von Fake News liegt in ihrer Aufmachung: Häufig wird die zentrale Aussage bereits in der Überschrift formuliert, oft mit Bezug auf konkrete Personen oder Ereignisse. Dadurch überspringen viele Leser den Artikel selbst, was die Wirkung von Schlagzeilen verstärkt. Die Inhalte sind meist kurz, wiederholend und wenig informativ. Anders als bei seriösen Nachrichten, bei denen Argumente überzeugen sollen, wirken Fake News über einfache Denkabkürzungen (Heuristiken) und die Bestätigung bestehender Überzeugungen. Nutzer müssen sich also nicht mit komplexen Inhalten auseinandersetzen, sondern lassen sich durch intuitive Übereinstimmungen überzeugen. Besonders bei geringer kognitiver Anstrengung – etwa durch Müdigkeit oder Unaufmerksamkeit – steigt die Wahrscheinlichkeit, dass Fake News geglaubt und weiterverbreitet werden [**?**].

1.1.5 Wer konsumiert Fake News

Laut [**?**] sind folgende Gruppen die größten Konsumenten:

- **Geringe Bildung oder digitale Kompetenz:** Personen mit niedriger formaler Bildung oder unzureichenden digitalen Fähigkeiten sind anfälliger für Falschinformationen.

- **Aussagen oder Nähe zur Informationsquelle:** Informationen von Personen, denen man persönlich nahe steht oder vertraut, werden eher geglaubt – unabhängig vom Wahrheitsgehalt.
- **Parteizugehörigkeit oder politische Überzeugung:** Menschen neigen dazu, Fake News zu glauben und zu verbreiten, wenn diese mit ihrer ideologischen Einstellung übereinstimmen.
- **Misstrauen gegenüber den Medien:** Wer etablierten Medien nicht vertraut, ist eher bereit, alternative (oftmals falsche) Quellen zu konsumieren und zu verbreiten.
- **Geringere kognitive Fähigkeiten:** Personen mit niedrigerer kognitiver Verarbeitungskapazität sind anfälliger für einfache, irreführende Inhalte und hinterfragen diese seltener kritisch.

Außerdem scheinen konservative, rechtsgerichtete Menschen, ältere Personen und weniger gebildete Menschen eher dazu zu neigen, Fake News zu glauben und zu verbreiten [**?**].

1.1.6 Welche potenziellen Indikatoren zum Erkennen bei Fake News gibt es

Das Erkennen von Fake News ist gerade deshalb problematisch, da diese erst erkannt werden können, nachdem sie erstellt und im Internet verbreitet wurden. [**?**]

Gerade im Bereich der sozialen Medien gibt es aber relativ zuverlässige Indikatoren, die Fake News nach der Erstellung als solche zu enttarnen [**?**]:

- **Fortlaufende Großschreibung:** Beispiel: `GROßSCHREIBUNG`
- **Übermäßige Nutzung von Satzzeichen:** Beispiel: `!!!`
- **Falsche Zeichensetzung am Satzende:** Beispiel: `!!1`
- **Übermäßige Nutzung von Emoticons, besonders auffälliger Emoticons**
- **Nutzung des Standard-Profilbildes**
- **Fehlende Account-Verifizierung, besonders bei prominenten Personen**

Fake News in offiziellen Nachrichtenportalen zu erkennen, ist dagegen deutlich schwieriger. Die aufgezählten stilistischen Mittel wie zum Beispiel die fortlaufende Großschreibung sind eher untypisch. Stattdessen muss über die inhaltliche Bedeutung erkannt werden ob die Artikel wahr oder falsch sind.

1.2 Zielsetzung: Entwicklung einer Software zur automatisierten Fake-News-Erkennung

Motiviert durch die Arbeiten der University of Applied Sciences Upper Austria [**?**] und der TU Darmstadt [**?**] wird in dieser Arbeit die Entwicklung eines weiteren Tools dokumentieren. Dieses Tool soll wie auch das Browser Plugin TrustyTweet eine Unterstützung zum Erkennen von Fake News anbieten. Ob dieses Tool auch als Browser Plugin oder als eine andere Form der Software implementiert wird, steht zum jetzigen Zeitpunkt noch nicht fest. Auch ob eine Black Box oder White Box Architektur genutzt wird, - das heißt, kann der User zum Beispiel sehen, warum der Artikel als Fake News deklariert wird - wird im Laufe der Arbeit entschieden. Ziel ist es, dass das Tool nicht wie TrustyTweet auf Twitter eingesetzt wird, sondern auf verschiedenen Nachrichtenportalen. Um eine politisch möglichst breite Abdeckung zu decken, wird das Tool für drei verschiedenen Nachrichtenportalen implementiert (siehe Kapitel 1.3).

1.3 Wahl der Nachrichtenportale

Im Paper der University of Applied Sciences Upper Austria [**?**] wird die Qualität verschiedener deutscher Nachrichtenportale mit „Machine Learning"-Modellen getestet. Das Ergebnis zeigt, dass Spiegel, Die Zeit and Süddeutsche die 'besten' Portale sind. Express, BZ-Berlin und Bild sind die 'schlechtesten', da sie am meisten Fake News verbreiten. Die Quellen [**? ? ?**] zeigen, dass die Kombination von BILD, taz und Der Spiegel eine politisch breites Spektrum abbilden.

- **BILD:** Boulevardesk, populistisch, konservativ

  Die BILD-Zeitung gilt als stark meinungsgetriebenes Boulevardmedium mit populistischen Zügen. Ihre Berichterstattung ist geprägt von einer emotionalisierenden Sprache, Fokus auf Einzelereignisse und dem Ziel hoher Reichweiten.

- **taz:** Kritisch, linksalternativ, bewegungsnah

  Die taz (tageszeitung) wird dem linksalternativen Spektrum zugeordnet. Sie verfolgt eine aktivistische Grundhaltung mit einem Fokus auf sozialen Bewegungen, Umweltfragen und Minderheitenrechten. Die taz gilt als Gegenmodell zu großen Leitmedien und strebt oft bewusst Gegenöffentlichkeit an.

- **Der Spiegel:** Linksliberal, investigativ, kritisch gegenüber Macht

  Der Spiegel wird dem linksliberalen Spektrum zugeordnet. Er kombiniert klassische Leitmedienformate mit einem ausgeprägten Anspruch auf investigativen Journalismus, Kritik an staatlicher Macht und liberal-demokratischen Werten.

Das Tool, welches in dieser Arbeit entwickelt wird, wird also für diese drei Nachrichtenportale implementiert.

1.4 Aufbau der Arbeit

2 Grundlagen und Begriffsdefinitionen

2.1 Definition „Fake News“: Merkmale, Ziele, Beispiele

2.1.1 Klassifizierungen

Linguistische Features werden von Textmaterial auf verschiedenen Leveln gesammelt, z.B. Buchstaben, Wörter, Sätze und Features auf dem Satzlevel (Häufigkeit von Funktionswörtern? und Sätzen) [**?**]

Text Tokenisierung [**?**]

Aufteilung der Features in Syntactic, Semantic, Sentiment, Lexical, Style-based [**?**] p7

2.2 Kategorisierung der Fake News Detection-Ansätze

2.3 Warum der Fokus auf Machine Learning?

2.4 Überblick über relevante Plattformen und deren Rolle im Medienkonsum

3 Natural Language Processing

3.1 Machine Learning

3.1.1 Textbereinigung und Vorverarbeitung

- **Titel und Inhalt der Artikel zusammenfügen [?]**: Damit keine wichtigen Informationen verloren gehen, werden Titel und Inhalt des Artikels zusammengefasst. Gerade der Titel kann durch z.B. Clickbait (siehe 1.1.2) schnell Hinweise auf eventuelle Fake News geben.
- **Akzente und Sonderzeichen entfernen [?] [?] [?]**: Akzente führen dazu, dass Wörter wie „café“ und „cafe“ unterschiedlich behandelt werden, obwohl sie semantisch gleich sind. Das Entfernen dieser erhöht die Generalisierung. Sonderzeichen stören einfache Tokenizer (z. B. bei Bag-of-Words), führen zu vielen seltenen Tokens und zu überdimensionierten Vektoren (siehe 3.1.2).
- **Alle Buchstaben zu Kleinbuchstaben konvertieren [?] [?] [?]**: Ähnlich wie zum vorherigen Punkt erhöht die durchgehende Kleinschreibung aller Buchstaben die Generalisierung und verhindert somit unnötige Duplikate im Vokabular.
- **Leere Spalten entfernen [?]**: Leere Spalten enthalten keine Information. Sie können bei der Vektorisierung oder Modellerstellung Fehler verursachen und werden als einfache Datenbereinigungsmaßnahme entfernt.
- **Kontraktionen auflösen (ans -> an das) [?]**: Im deutschen sind Kontradiktionen zwar nicht so häufig wie im englischen, sie kommen aber trotzdem vor und sollten aufgelöst werden. Dies vermeidet fragmentierte Token und verbessert die Semantik und Trennbarkeit im Modell.

- **Stoppwörter entfernen [?] [?] [?]**: Wörter wie „der“, „ist“, „und“ tragen wenig zur inhaltlichen Differenzierung bei. Das Entfernen dieser verbessert die semantische Gewichtung relevanter Begriffe [**?**].
- **Rechtschreibfehler korrigieren [?]**: Tippfehler führen zu seltenen Tokens und stören die Generalisierung. In offiziellen Artikeln sind zwar selten Rechtschreibfehler zu finden, aber falls vorhanden, hilft die Korrektur zur Verbesserung der Modellqualität.
- **Lemmatisieren [?] [?] [?]**: Bei der Lemmatisierung werden verschiedene Wortformen auf die Grundform zurückgeführt („läuft“, „lief“, „laufen“ wird zu „laufen“). So erkennt das Modell gleiche Bedeutungen trotz grammatischer Variation.
- **Tokenisierung [?]**: In der Tokenisierung werden die Texte in einzelne Wörter oder Einheiten (Tokens) zerlegt, die für Modelle verarbeitbar sind. Dies ist eine Grundvoraussetzung für alle weiteren NLP-Schritte wie TF-IDF oder Word Embeddings.

Nutzung einer duale Feature-Pipeline

Ein Problem welches das Entfernen der Akzente und Sonderzeichen und das Konvertieren aller Buchstaben zu Kleinbuchstaben mit sich bringt ist, dass viele wichtige Hinweise zum Erkennen von Fake News verloren gehen. Wie in Kapitel 1.1.6 beschrieben, sind fortlaufende Großschreibung, übermäßige Nutzung von Satzzeichen und falsche Zeichensetzung am Satzende potenzielle Indikatoren für Fake News.

Eine duale Feature-Pipeline kann dieses Problem lösen. Implementiert wird eine „cleaned“ Version (z.B. für inhaltliche Bedeutung) mit standardisierten, inhaltlichen Features und eine „rohe“ Version (z.B. für Stilmerkmale) mit stilistischen, rohen Textfeatures.

So werden semantische und stilistische Hinweise genau so genutzt wie ein Mensch es beim Lesen macht.

Die Notwendigkeit der Stilmerkmale ist aber diskutierbar. Die Datensätze werden ausschließlich aus Artikeln von offiziellen Nachrichtenportalen zusammengesetzt. Diese schreiben meist sauber, ohne Caps-Lock oder auffällige Sonderzeichen. Stilistische Merkmale wie viele Ausrufezeichen, Emojis oder absichtliche Rechtschreibfehler kommen dort nicht vor – also sind sie in diesem Fall auch keine verlässlichen Fake-News-Signale.

3.1.2 Merkmalextraktion

Bag-of-words

Das Bag-of-Words-Modell ist ein einfaches Verfahren zur Textrepräsentation, bei dem ein Dokument als Vektor der Häufigkeiten einzelner Wörter dargestellt wird – unabhängig von deren Reihenfolge oder Kontext. Es zählt lediglich das Vorkommen jedes Wortes aus einem festen Vokabular [**?**].

TF-IDF

TF-IDF ist ein gewichtetes Modell zur Textdarstellung, das berücksichtigt, wie häufig ein Wort in einem Dokument vorkommt (TF) und wie selten es im gesamten Kontext ist (IDF). Es dient dazu, häufige, aber wenig informative Wörter zu reduzieren und aussagekräftige Begriffe zu betonen [**?**].

Sparse Matrizen werden sowohl von Bag-of-Words als auch von TF-IDF genutzt. Eine Matrix wird als sparse bezeichnet, wenn der Anteil der Nicht-Null-Werte im Verhältnis zur Gesamtanzahl der Dokumente sehr klein ist. Pro hinzugefügtem Dokument wird eine Zeile erstellt und pro Wort im Vokabular eine Spalte. Da jedes Dokument nur einen Bruchteil der Wörter des Gesamtvokabulars enthält, bestehen der Großteil einer solchen Matrix aus Nullen.

	Doc 1	Doc 2	Doc 3
baking	0	1	1
cake	1	1	1
chocolate	1	0	0
he	0	0	1
her	0	0	1
is	0	1	1
loves	1	0	0
she	1	1	0
surprise	0	0	1
to	0	0	1

(a) Bag-of-words Sparse Matrix [**?**]

	Doc 1	Doc 2	Doc 3
baking	0	0.52	0.32
cake	0.34	0.40	0.25
chocolate	0.58	0	0
he	0	0	0.42
her	0	0	0.42
is	0	0.52	0.52
loves	0.55	0	0
she	0.44	0.52	0
surprise	0	0	0.42
to	0	0	0.42

(b) TF-IDF Sparse Matrix [**?**]

Abbildung 3.1: Vergleich der Sparse Matrizen

In Abbildung 3.1 wurden den beiden Matrizen jeweils die drei Dokumente:

- Doc1 - She loves chocolate cake
- Doc2 - She is baking a cake
- Doc3 - He is baking a cake to surprise her

hinzugefügt. In der Matrix 3.1a werden in jeder Zelle in welcher das Dokument das entsprechende Wort beinhaltet eine 1 gesetzt. In 3.1b wird statt einer 1 eine Gewichtung über die Häufigkeit der Wörter in allen Dokumenten hinweg erstellt und eingetragen. Sie bewertet die Wichtigkeit eines Wortes in einem Dokument relativ zur gesamten Sammlung von Dokumenten. Dabei wird die Termfrequenz (TF) mit der invertierten Dokumentfrequenz (IDF) multipliziert. Je höher der resultierende Wert, desto relevanter ist das Wort für das jeweilige Dokument. Die Formel lautet:

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D) \tag{3.1}$$

Dabei ist t das Wort, d das Dokument und D die gesamte Dokumentensammlung [**?**].

Die TF misst, wie häufig ein bestimmter Begriff t in einem Dokument d vorkommt. Sie beschreibt die lokale Bedeutung eines Wortes innerhalb des Dokuments.

$$\text{tf}(t, d) = \frac{\text{Anzahl der Vorkommen von } t \text{ in } d}{\text{Gesamtanzahl der Wörter in } d} \tag{3.2}$$

Die IDF bewertet, wie selten ein Begriff t in der gesamten Dokumentensammlung D ist. Je seltener ein Begriff in vielen Dokumenten vorkommt, desto höher ist sein IDF-Wert.

$$\text{idf}(t, D) = \log\left(\frac{N}{\text{df}(t)}\right) \tag{3.3}$$

Dabei ist N die Gesamtanzahl der Dokumente in der Matrix und $\text{df}(t)$ die Anzahl der Dokumente, in denen der Begriff t vorkommt [**?**].

Die in [**?**] beschriebene Relevance Frequency (RF) ist eine überwachte Gewichtungsform der IDF-Komponente im TF-IDF, die nicht nur zählt, in wie vielen Dokumenten ein Begriff vorkommt, sondern berücksichtigt, in welchen Klassen der Begriff besonders häufig oder exklusiv ist. Die Formel lautet:

$$\mathrm{rf}(t) = \log\left(2 + \frac{P(t)}{\max(1, N(t))}\right) \tag{3.4}$$

Mit $P(t)$ für die Anzahl der relevanten Dokumente (z. B. positive Klasse), in denen der Term t und $N(t)$ für die Anzahl der irrelevanten Dokumente (z. B. negative Klasse), in denen der Term t vorkommt.

Während klassisches IDF ein Wort umso höher gewichtet, je seltener es allgemein in der Gesamtmatrix ist, gewichtet RF hingegen ein Wort umso höher, je stärker es mit einer bestimmten Zielklasse assoziiert ist. Dadurch hebt RF Begriffe hervor, die klassenunterscheidend sind was beim Arbeiten mit überwachten Modellen relevant ist.

In der IF-IDF wird für IDF wird nun also RF eingesetzt und es ergibt sich folgende Formel:

$$\mathrm{tfidf}(t, d) = \frac{\text{Anzahl der Vorkommen von } t \text{ in } d}{\text{Gesamtanzahl der Wörter in } d} \cdot \log\left(2 + \frac{P(t)}{\max(1, N(t))}\right) \tag{3.5}$$

- Mit dem Wort t und dem Dokument d
- $P(t)$ für die Anzahl der relevanten Dokumente (z. B. positive Klasse), in denen der Term t vorkommt
- $N(t)$ für die Anzahl der irrelevanten Dokumente (z. B. negative Klasse), in denen der Term t vorkommt

Vergleich Bag-of-words und TF-IDF

Aus Tabelle 3.1 zu erkennen ist, dass TF-IDF in vielen Anwendungen leistungsfähiger ist als BoW. Insbesondere bei Texten mit hohem Vokabularumfang.

Hashing Vectorizer

Ein Hashing Vectorizer ist eine Methode zur Umwandlung von Text in numerische Merkmalsvektoren, ohne dass ein Vokabular explizit erstellt oder gespeichert wird. Stattdessen wird eine Hash-Funktion verwendet, um jedes Wort auf einen Index im Feature-Vektor abzubilden [**?**].

Bag-of-Words (BoW)	**TF-IDF**
Einfache Implementierung [?]	Berücksichtigt Wortwichtigkeit in gesamter Matrix [?]
Keine Gewichtung — häufige Wörter dominieren	Seltener, aber informativer Inhalt wird stärker gewichtet [?]
Hohe Dimensionalität in Sparse Matrix (jedes Wort bekommt eine separate Dimension) [?]	Gleiches Problem, aber mit informativeren Werten [?]
Ignoriert Wortreihenfolge und Kontext [?]	Gleiches Grundproblem, aber geringfügig bessere Performance [?]
Nützlich für einfache Klassifikatoren	Bessere Klassifikationsergebnisse in Kombination mit SVM oder Logistic Regression [?]

Tabelle 3.1: Vergleich der Vor- und Nachteile von BoW und TF-IDF

In Abbildung 3.2 wird der Vergleich zwischen Machine-Learning-Modellen unter Verwendung von BoW-, TF-IDF- und Hashing-Features gezeigt. Die y-Achse präsentiert den F1-Score, also die Precision und Recall-Werte der jeweiligen Modelle. Das Random-Forest-Modell (RF) zeigt eine schwache Leistung bei der Verwendung von Hashing, während die linearen Modelle (z.B. SVM (Support Vector Machine) und LR (Logistische Regression)) ihre F1-Werte mit Hashing-Features verbessern konnten.

Die Verbesserung ist aber nur minimal. Der F1-Score bei SVM ist ohne Hashing bei 0.89 und nach bei 0.90. Bei LR steigt der Wert auch von 0.87 und 0.88.

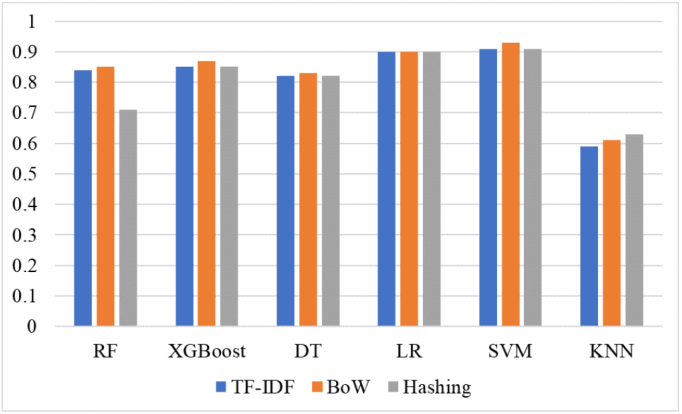

Abbildung 3.2: Vergleich verschiedener Modelle mit BoW, TF-IDF und Hashing [?]

3.1.3 Machine Learning Modelle

Naive-Bayes

Der Naive-Bayes-Algorithmus ist ein einfacher, aber leistungsfähiger Klassifikator, der auf dem Satz von Bayes basiert. Er wird häufig in Bereichen wie Textklassifikation, Spam-Erkennung und Sentiment-Analyse eingesetzt [**?**].

Der Klassifikator nutzt den Satz von Bayes zur Berechnung der Wahrscheinlichkeit einer Klasse C gegeben eine Merkmalsmenge X:

$$P(C \mid X) = \frac{P(X \mid C) \cdot P(C)}{P(X)} \tag{3.6}$$

Dabei ist:

- $P(C \mid X)$ – *Posterior*: Die Wahrscheinlichkeit für die Klasse C, nachdem die Daten X beobachtet wurden.
- $P(X \mid C)$ – *Likelihood*: Die Wahrscheinlichkeit, die Daten X zu sehen, wenn sie zur Klasse C gehören.
- $P(C)$ – *Prior*: Die ursprüngliche Wahrscheinlichkeit der Klasse C, ohne Kenntnis über die Daten.
- $P(X)$ – *Evidenz*: Die Gesamtwahrscheinlichkeit, die Daten X zu beobachten (über alle Klassen hinweg).

Die zentrale Annahme des Naive-Bayes-Klassifikators ist die bedingte Unabhängigkeit der Merkmale:

$$P(X \mid C) = \prod_{i=1}^{n} P(x_i \mid C) \tag{3.7}$$

Dies vereinfacht die Berechnung erheblich, da nur die Wahrscheinlichkeiten einzelner Merkmale betrachtet werden müssen [**?**].

Decision Tree

Ein Decision Tree (Entscheidungsbaum) ist ein Algorithmus für Klassifikation und Vorhersage. Er basiert auf einer baumartigen Struktur, bei der jeder Knoten bzw. Ast ein Merkmal aus einem Datensatz repräsentiert. Diese Struktur ermöglicht es, schrittweise Entscheidungen zu treffen, die schließlich zu einer Klassenzuordnung an einem Blattknoten führen [**?**].

Der Baum wird durch Auswahl von Merkmalen aufgebaut, die die Daten am besten aufspalten. Dieses Auswahlkriterium basiert auf dem Konzept der **Entropie** und dem daraus abgeleiteten **Informationsgewinn**. Ziel ist es, bei jeder Entscheidung im Baum das Merkmal auszuwählen, das die größte Reduktion an Unsicherheit bietet.

Die Entropie misst die Unreinheit oder Unbestimmtheit eines Datensatzes. Sie ist dann maximal, wenn alle Klassen gleichverteilt sind, und minimal (d.h. null), wenn alle Daten zur selben Klasse gehören. Die Entropie $E(S)$ eines Datensatzes S wird wie folgt berechnet:

$$E(S) = -\sum_{i=1}^{c} p_i \log_2 p_i \tag{3.8}$$

Dabei ist:

- c die Anzahl der Klassen,
- p_i der Anteil der Klasse i im Datensatz S.

Der Informationsgewinn misst die Reduktion der Entropie, die durch das Aufteilen eines Datensatzes mittels eines bestimmten Merkmals erzielt wird. Je größer der Informationsgewinn, desto besser ist das Merkmal für die Aufspaltung geeignet. Eine alternative Formel für den Informationsgewinn $IG(E)$ lautet:

$$IG(E) = 1 - \sum_{i=1}^{c} p_i^2 \tag{3.9}$$

Über einen Hyperparameter kann die maximale Tiefe des Baumes festlegt werden. Eine zu große Tiefe kann zu Overfitting führen, da der Baum zu sehr an die Trainingsdaten angepasst wird [**?**].

Random Forest

Ein Random Forest besteht aus einer großen Anzahl von Entscheidungsbäumen. Jeder Baum wird auf einem zufällig gezogenen Teildatensatz trainiert (Bagging). Bei der Bildung jedes Knotens (Split) wird eine zufällige Teilmenge von Merkmalen berücksichtigt. Die finale Klassifikation ergibt sich durch Mehrheitsentscheidung aller Bäume (Ensemble Voting) [**?**].

Die Bedeutung eines Merkmals i im Random Forest ergibt sich aus der durchschnittlichen normierten Bedeutung dieses Merkmals über alle Entscheidungsbäume hinweg. Diese kann mathematisch wie folgt dargestellt werden:

$$RFf_i = \frac{\sum_{j \in \text{all trees}} normf_{ij}}{T} \tag{3.10}$$

Dabei ist:

- RFf_i die Gesamtrelevanz der Klasse i im gesamten Wald,
- $normf_{ij}$ die normierte Wichtigkeit des Merkmals i im Baum j,
- T die Gesamtanzahl der Entscheidungsbäume [**?**].

Wichtige Hyperparameter sind hier:

- `n_estimators`: Anzahl der Entscheidungsbäume im Wald.
- `max_depth`: Maximale Tiefe der Bäume.
- `max_features`: Anzahl der Merkmale, die für einen Split berücksichtigt werden.
- `bootstrap`: Gibt an, ob Stichproben mit Zurücklegen gezogen werden.

Im Vergleich zu Decision Trees ist Random Forest robuster gegenüber Overfitting und bringt durch das Ensemble Voting eine höhere Genauigkeit [**?**].

Support Vector Machines

Support Vector Machines (SVMs) sind überwachte Lernalgorithmen, die besonders effektiv für Klassifikationsaufgaben sind. Sie finden breite Anwendung in Bereichen wie Bioinformatik, Textklassifikation und insbesondere in der Erkennung von Fake News. Das Ziel einer SVM ist es, eine Trennlinie — oder in höherdimensionalen Räumen ein Trenn-Hyperplane — zu finden, das Datenpunkte verschiedener Klassen mit maximalem Abstand (Margin) voneinander trennt [**? ? ? ?**].

1. Ziel: Eine Trennebene finden Eine SVM sucht eine Gerade (in 2D), Ebene (in 3D) oder Hyperplane, die die Klassen voneinander trennt:

$$w^T x + b = 0 \tag{3.11}$$

2. Bedingung für korrekte Trennung Für jeden Punkt x_i mit zugehörigem Label $y_i \in \{-1, +1\}$ gilt:

$$y_i(w^T x_i + b) \geq 1 \tag{3.12}$$

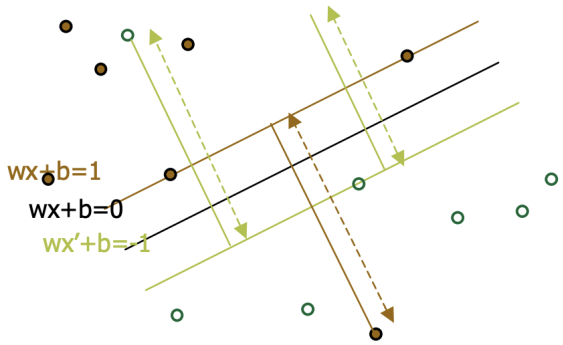

Abbildung 3.3: Darstellung von Hyperplanes [**?**]

3. Margin maximieren Der Margin ist der Abstand der nächsten Punkte beider Klassen zur Trennebene:

$$M = \frac{2}{\|w\|} \tag{3.13}$$

Daher wird zur Maximierung des Margins folgende Zielfunktion minimiert:

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 \quad \text{mit} \quad y_i(w^T x_i + b) \geq 1 \tag{3.14}$$

4. Fehler zulassen – Soft Margin

Bei nicht perfekt trennbaren Daten werden sogenannte Slack-Variablen ξ_i eingeführt:

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \tag{3.15}$$

Ziel ist es, Fehler klein zu halten, gleichzeitig aber den Margin möglichst groß:

$$\min \left(\frac{1}{2}\|w\|^2 + C \sum \xi_i \right) \tag{3.16}$$

5. Nichtlineare Trennung – Kernel-Trick

Bei komplexen Datensätzen wird das Problem durch eine Funktion ϕ in höhere Dimensionen überführt, ohne sie explizit zu berechnen [**?**]:

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \tag{3.17}$$

Gängige Kernel-Funktionen:

- Linearkernel: $K(x, x') = x^T x'$
- Polynomial: $K(x, x') = (x^T x' + 1)^d$
- Radial Basis Function (RBF): $K(x, x') = \exp(-\gamma \|x - x'\|^2)$ [**?**]

[**?**] zeigt, dass durch geeignete Wahl eines Kernels auch komplex strukturierte Daten erfolgreich klassifiziert werden können.

Es ergibt sich folgender Ablauf:

1. Definiere Trennebene: $w^T x + b = 0$
2. Erzwinge korrekte Trennung: $y_i(w^T x_i + b) \geq 1$
3. Maximiere Margin: $\min \frac{1}{2}\|w\|^2$
4. Erlaube kleine Fehler: $\min \frac{1}{2}\|w\|^2 + C \sum \xi_i$
5. Wende ggf. Kernel an: $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$

Vorteile von SVMs sind:

- **Robustheit gegenüber Overfitting**, insbesondere bei hoher Dimensionalität und geringem Datensatz [**?**].
- **Gute Generalisierungsfähigkeit** durch Maximierung des Margins.
- **Effizient in der Praxis**: Für viele reale Probleme sind SVMs konkurrenzfähig gegenüber tieferen Netzwerken, (z.B. in Fake-News-Klassifikationen) [**? ?**].
- **Flexibilität durch Kernel-Funktionen**, womit verschiedene Datentypen (z.B. Vektoren, Strings, Graphen) verarbeitet werden können.

Logistische Regression

Die logistische Regression (LR) ist ein weit verbreitetes Verfahren des überwachten maschinellen Lernens zur Klassifikation binärer und multiklassiger Zielvariablen. Sie wird eingesetzt, um die Wahrscheinlichkeit zu berechnen, mit der eine Beobachtung zu einer bestimmten Klasse gehört [**? ? ?**]. Im Gegensatz zur linearen Regression verwendet LR eine Aktivierungsfunktion, typischerweise die Sigmoidfunktion, um Ausgaben zwischen 0 und 1 abzubilden. Diese Werte stellen Wahrscheinlichkeiten dar und werden zur Vorhersage diskreter Zielwerte genutzt [**?**]. Die Tabelle 3.2 zeigt die Aktivierungsfunktionen, mit deren entsprechend benötigten Zielvariablen.

Typ	**Aktivierungsfunktion**	**Typ der Zielvariable**
Binäre logistische Regression	Logit (Sigmoid): $\frac{1}{1+e^{-z}}$	Binär (0/1)
Multinomiale logistische Regression	Softmax: $\frac{e^{z_k}}{\sum_j e^{z_j}}$	Kategorisch (mehrere Klassen)
Ordinale logistische Regression	Cumulative Logit, Probit, Cloglog	Geordnete Klassen
Probit-Modell	$\Phi(z)$ (Normalverteilung)	Binär (0/1), robust gegen Ausreißer

Tabelle 3.2: Übersicht von Aktivierungsfunktion in der logistischen Regression [**? ? ?**]

Das Ziel der logistischen Regression ist es, eine Funktion zu finden, die die Wahrscheinlichkeit P berechnet, dass ein Eingabewert X zur Klasse $y = 1$ gehört. Dies geschieht zum Beispiel mittels der Sigmoidfunktion [**?**]:

$$P = \frac{1}{1 + e^{-(a+bX)}} \tag{3.18}$$

Dabei sind:

- P: Wahrscheinlichkeit für Klasse 1 (Wert zwischen 0 und 1)
- X: Eingabewert (Merkmalsvariable)
- b: Gewicht des Merkmals
- a: **Bias** oder **Intercept**, also der Schnittpunkt mit der y-Achse, verschiebt die Entscheidungsgrenze. Ohne Bias würde die Entscheidungsgrenze immer durch den Ursprung laufen, was in der Praxis selten sinnvoll ist [**?**].

In [**?**] wurde der Intercept explizit deaktiviert, wodurch sich die Gleichung zu $P = 1/(1 + e^{-bX})$ vereinfacht.

Vorteile der logistischen Regression:

- **Einfachheit und Interpretierbarkeit**: LR-Modelle sind leicht verständlich und liefern direkt interpretierbare Wahrscheinlichkeiten [**?**].
- **Effizienz**: Sie sind schnell trainierbar und benötigen relativ geringe Rechenleistung [**?**].
- **Flexibilität**: LR lässt sich für binäre, multinomiale und ordinale Klassifikationsprobleme erweitern [**?**].
- **Breite Anwendbarkeit**: Sie wird in zahlreichen Bereichen eingesetzt, von der Medizin bis zur Textklassifikation [**? ?**].

K-Nearest Neighbor

Der K-Nearest Neighbor (KNN) Algorithmus ist ein überwachter Lernalgorithmus, der unter anderem für Klassifikationsprobleme eingesetzt wird. Er gehört zur Gruppe der sogenannten lazy learners, da kein expliziter Trainingsprozess stattfindet. Stattdessen wird der Trainingsdatensatz während der Vorhersage verwendet [**?**].

KNN klassifiziert neue Instanzen auf Basis ihrer Ähnlichkeit mit bereits bekannten Beispielen. Dazu berechnet es die Distanz zwischen dem Testpunkt und allen Trainingspunkten. Anschließend werden die K ähnlichsten Punkte ausgewählt und die Vorhersage erfolgt bei Klassifikation durch Mehrheitsentscheidung [**?**].

Die Distanzmessung erfolgt in der Regel über die **euklidische Distanz**, mit der gemessen wird, wie weit zwei Punkte im Merkmalsraum voneinander entfernt sind. Die Formel lautet:

$$E_d = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2} \tag{3.19}$$

Dabei sind:

- x_i: der i-te Wert des zu klassifizierenden Punkts
- y_i: der entsprechende i-te Wert eines bekannten Trainingspunkts
- k: die Anzahl der Merkmale (Dimensionen), z.B. bei Textdaten die Länge des Vektors

Je kleiner der Wert E_d, desto ähnlicher sind sich die beiden Punkte.

Der KNN-Algorithmus verwendet mehrere wichtige Hyperparameter:

1. **n_neighbors**: Gibt an, wie viele der nächsten Trainingspunkte zur Klassifikation berücksichtigt werden. Der Wert (oft als K bezeichnet) sollte basierend auf den Eigenschaften des Datensatzes gewählt werden [**?** **?**].
2. **weights**: Bestimmt, ob allen Nachbarn das gleiche Gewicht gegeben wird oder ob näher gelegene Punkte stärker gewichtet werden [**?**].
3. **metric**: Die Distanzmetrik zur Berechnung der Ähnlichkeit. Standardmäßig wird die euklidische Distanz verwendet, möglich sind aber auch Manhattan, Minkowski oder andere Metriken [**?**]

Auch wenn KNN einfach und intuitiv zu implementieren ist und keinen Trainingsprozess benötigt, zeigt Abbildung 3.4, dass es im Vergleich mit Naive Bayes (NB), logistischer Regression (LG) und Support Vector Machines (SVM) ein vergleichsweise ineffektives Model für NLP ist.

Hier wurde KNN zur Klassifikation in einem Stimmungserkennungs-System für die Amtssprache Kambodschas eingesetzt. Dabei diente KNN dem Vergleich mit anderen Ansätzen wie SVM und wurde insbesondere hinsichtlich seiner Leistung bei der Klassifikation von

Approach	Accuracy	Precision	Recall	F1-Score
NB	73%	73%	73%	73%
K-NN	79%	75%	79%	77%
RF	81%	78%	81%	78%
SVM	85%	81%	85%	83%
BiLSTM	86%	84%	86%	84%

Abbildung 3.4: Auswertung verschiedener NLP Algorithmen [**?**]

Textdaten bewertet, bei denen die Reihenfolge, der Zeitpunkt oder der Verlauf über die Zeit eine wichtige Rolle für die Interpretation und Analyse spielt.

XGBoost

Beim XGBoost (eXtreme Gradient Boosting) wird das Modell durch die Addition mehrerer Entscheidungsbäume aufgebaut, welche als schwache Lernalgorithmen (base learners) fungieren. Anders als bei Random Forests, bei denen Bäume unabhängig voneinander trainiert und aggregiert werden, lernen die Bäume in XGBoost aufeinander aufbauend (siehe Abbildung 3.5). Die Vorhersage für ein Beispiel ergibt sich aus der Summe der Ausgaben aller zuvor gelernten Bäume. Dadurch entsteht ein starkes Modell, das schrittweise durch Fehlerkorrektur verbessert wird [**? ? ?**].

Ein zentraler Vorteil von XGBoost ist die integrierte Regularisierung, mit der das Modell Overfitting vermeiden kann. Dabei werden zwei Arten von Regularisierung eingesetzt:

- **L1-Regularisierung**: Bestraft große Gewichtswerte, indem sie einige Gewichte auf Null setzt. Dadurch hilft sie, unwichtige Merkmale automatisch zu entfernen.
- **L2-Regularisierung**: Bestraft extreme Gewichtswerte, ohne sie komplett zu eliminieren. Dies führt zu stabileren Modellen mit kleinen, gleichmäßigen Gewichten.

Beide Regularisierungen sind in die sogenannte Ziel- oder Kostenfunktion eingebettet, die das Modell bei jedem Trainingsschritt minimiert.

[1] https://flower.ai/blog/2023-11-29-federated-xgboost-with-bagging-aggregation/

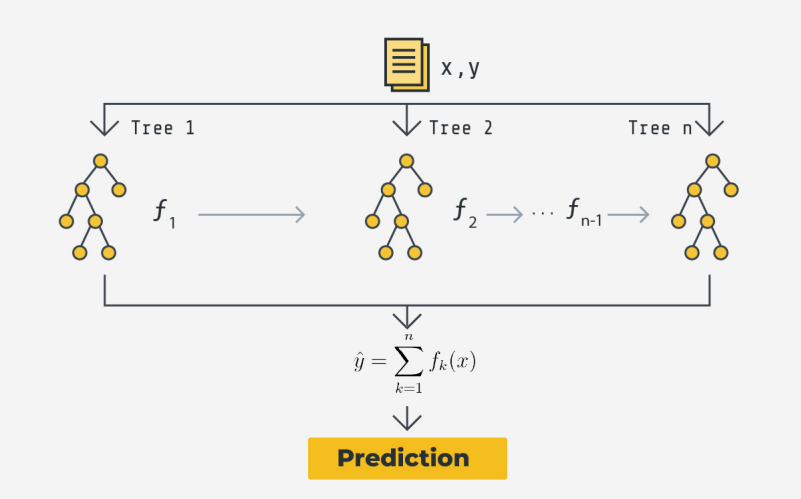

Abbildung 3.5: XGBoost[1]

In Anwendungen der natürlichen Sprachverarbeitung (NLP) hilft diese Kombination, besonders bei großen Textmerkmalräumen (z.B. TF-IDF), relevante Merkmale herauszufiltern und gleichzeitig stabile Modelle zu trainieren[**?**].

3.1.4 Metriken

Zur Auswertung der überwachten Modelle werden Metriken genutzt. Die Auswahl der richtigen Metriken hängt von der gewünschten Zielsetzung ab. Diese kann zum Beispiel binäre, bzw. multi-Klassen Klassifikation oder Regression sein.

Fake News Erkennung ist eine binäre Klassifizierung (Der Artikel ist entweder 'wahr' oder 'falsch').

Dabei ergeben sich vier mögliche Ausgänge bei der Modellvorhersage:

- True Positive (TP) - Das Modell hat die positive Klasse vorhergesagt.

  (Der Artikel, der kein Fake ist, wird als 'wahr' gedeutet)

- True Negative (TN) - Das Modell hat die negative Klasse richtig vorhergesagt.

  (Der Artikel, der Fake ist, wird als 'falsch' gedeutet)

- Falsch positiv (FP) - Das Modell hat die positive Klasse falsch vorhergesagt.

  (Der Artikel, der kein Fake ist, wird als 'falsch' gedeutet)

- Falsches Negativ (FN) - Dein Modell hat die negative Klasse falsch vorhergesagt.

  (Der Artikel, der kein Fake ist, wird als 'wahr' gedeutet)

Diese vier Werte werden in einer sogenannten Konfusionsmatrix (siehe Abbildung 3.6) zusammengefasst aus der verschiedene Bewertungsmetriken abgeleitet werden können.

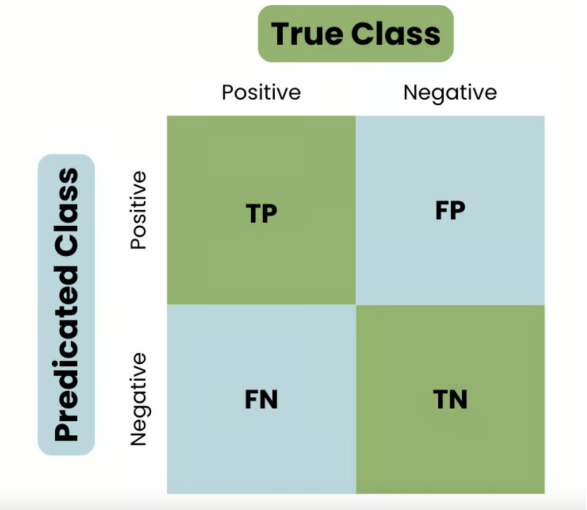

Abbildung 3.6: Konfusionsmatrix[2]

Nach [**?** **?**] sind die relevantesten Metriken für binäre Klassifikationen Accuracy, Recall (Sensitivity in [**?**]), Specificity und Precision.

Accuracy

Die Accuracy gibt den Anteil korrekt klassifizierter Instanzen an.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.20}$$

[2] https://www.datacamp.com/de/tutorial/what-is-a-confusion-matrix-in-machine-learning

Recall

Der Recall gibt den Anteil korrekt erkannter positiver Fälle an.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.21}$$

Specificity

Die Specificity gibt den Anteil korrekt erkannter negativer Fälle an.

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{3.22}$$

Precision

Die Präzision gibt den Anteil tatsächlich positiver Fälle unter allen als positiv vorhergesagten Fällen an.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.23}$$

F1-Score

Der F1-Score vereint die beiden Metriken Recall und Precision in einem einzigen Wert und ist hilfreich, wenn ein Gleichgewicht zwischen diesen beiden wichtig ist – vor allem bei unausgeglichenen Datensätzen, bei denen Accuracy allein irreführend sein kann.

Sind in dem Datensatz der Fake News Erkennung zum Beispiel 95% der Artikel 'wahr' und 5% 'falsch' hat ein Modell das ausschließlich 'wahr' vorhersagt eine Accuracy von 95%. Es erkennt aber keinen einzigen Artikel der Fake ist. Der Recall wäre in diesem Fall 0 und somit auch der F1-Score.

$$\text{F1-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.24}$$

3.2 Deep Learning

3.2.1 Word Embeddings

Die klassischen Merkmalsextraktionen in Kapitel 3.1.2 eignen sich gut für klassische Machine Learning Modelle, wie Support Vector Machines oder Logistische Regression. Im Vergleich zu Word Embeddings erfassen diese aber keine semantischen Beziehungen. Word Embeddings verstehen die Bedeutung der einzelnen Wörter im gesamten Kontext [**?**] und repräsentieren dabei das ursprüngliche Wort in einem neuen Vektorraum, wobei aber die Eigenschaften des Wortes und seine Verbindungen zu anderen Wörtern bestmöglich bewahrt werden [**?**]. Dabei werden mit maschinellen Lerntechniken verschiedene dichte Vektoren mit einer festgelegten Dimension gebildet. Word Embeddings sind gegenüber zu BOW (Sparse Matrizen) deutlich speicherschonender.

Das Wort „Bank“ zum Beispiel hat in den Sätzen „Ich setze mich auf die Bank.“ und „Ich raube die Bank aus.“ zwei unterschiedliche Bedeutungen. Word Embeddings erkennt diese und erstellt für die zwei Kontexte/Wörter zwei verschiedene Vektoren [**?**].

In Abbildung 3.7 wird jedes Wort eines Korpus mit 6 Wörtern als dreidimensionaler Vektor dargestellt. Ziel von Word2Vec ist hierbei, dass Wörter mit ähnlichen Bedeutungen oder Kontexten ähnliche Vektordarstellungen haben. Die Ähnlichkeit der Vektoren „Katze“ und „Hund“ zeigt die semantische Beziehung zueinander. Die Vektoren „glücklich“ und „traurig“ hingegen zeigen in entgegengesetzte Richtungen, was auf ihre gegensätzlichen Bedeutungen hinweist [**?**].

Abbildung 3.7: Bsp. für Word Embeddings in einem dreidimensionalen Vektorraums [**?**]

Word2Vec

Das Word Embedding Word2vec verwendet ein neuronales Netzwerk und erfasst numerisch die Ähnlichkeiten zwischen Wörtern aufgrund ihrer kontextuellen Merkmale. Am häufigsten wird sie zur Analyse der semantischen Verbindungen zwischen Wörtern in einem Textkorpus eingesetzt [**?**].

Im Beispielsatz 'Mann verhält sich zu Frau wie König zu x.' erkennt Word2vec, dass für x = Königin gilt. Word2Vec löst solche Aufgaben, indem es alle Wörter x' im Gesamtvokabular V ausprobiert und das Wort findet, das folgende Gleichung maximiert [**?**]:

$$\hat{x} = \underset{x' \in V}{\operatorname{argmax}} \operatorname{sim}(x', \vec{\text{king}} + \vec{\text{woman}} - \vec{\text{man}}) \tag{3.25}$$

Wie in Abbildung 3.8 zu sehen, gibt es für Word2Vec zwei verschiedene Implementierungen. Im CBOW-Modell (continuous bag-of-words) wird ein Wort aufgrund seines Kontextes vorhergesagt. Im Skip-gram-Modell wird hingegen Kontexte aufgrund eines Wortes vorhergesagt.

Bei einem relativ kleines Korpus, empfiehlt Google aufgrund seiner ausgeprägten Fähigkeit mit niedrigfrequenten Wörtern zu arbeiten, das Skip-gram-Modell anzuwenden [**?**].

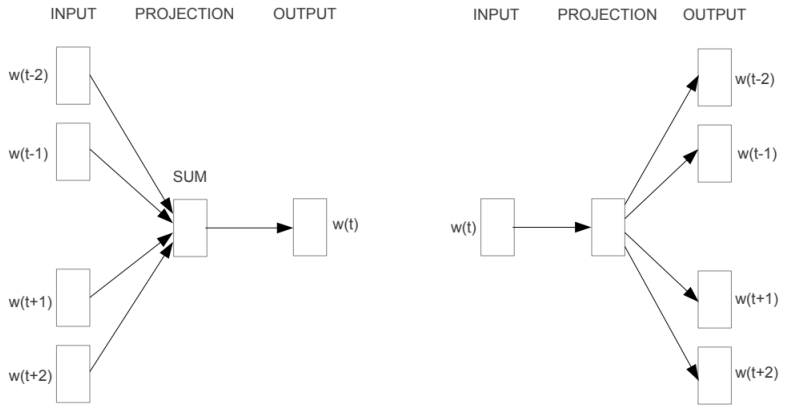

Abbildung 3.8: Vergleich CBOW und Skip-gram [**?**]

GloVe

Word2Vec fokussiert sich auf Informationen aus lokalen Kontextfenster, wobei globale Informationen hierbei nicht ausreichend genutzt werden. GloVe (Global Vectors for Word Representation) verwendet diese globalen Informationen, wodurch semantische Beziehungen zwischen Wörtern erfasst werden. Diese werden in einer globalen Co-Occurrence-Matrix zusammengefasst, die darstellt, wie oft Wörter zusammen im Korpus vorkommen [**?**].

Nach Erstellung der Matrix wird eine Zielfunktion formuliert, die die Beziehung zwischen den Wortvektoren und ihren Koinzidenzwahrscheinlichkeiten beschreibt .

Die Zielfunktion wird mithilfe von Gradientenabstieg oder anderen Optimierungsalgorithmen optimiert. Das Ziel ist es, die Wortvektoren und Verzerrungen so anzupassen, dass die quadratische Differenz zwischen der vorhergesagten und der tatsächlichen logarithmischen Wahrscheinlichkeit des gemeinsamen Auftretens minimiert wird [**?**].

Gegeben ein Korpus mit U Wörtern, dann ist die Kookkurrenzmatrix $X \in \mathbb{R}^{U \times U}$. Das Element X_{ij} steht für die Häufigkeit, mit der das Wort w_j im Kontext von w_i auftritt.

Die bedingte Wahrscheinlichkeit:

$$P(j|i) = \frac{X_{ij}}{X_i}$$

2. Verhältnis von Wahrscheinlichkeiten

Zur Modellierung semantischer Relationen vergleicht GloVe Wahrscheinlichkeitsverhältnisse:

$$\frac{P_{ik}}{P_{jk}} = \frac{X_{ik}/X_i}{X_{jk}/X_j}$$

Dies hilft zu beurteilen, ob w_i (z. B. „ice“) näher an k (z. B. „solid“) liegt als w_j (z. B. „gas“).

3. Vektorielle Darstellung

Um Vektoren und Wahrscheinlichkeitsverhältnisse zu verbinden, wird folgende Gleichung verwendet:

$$\vec{w}_i^T \vec{\tilde{w}}_j + b_i + \tilde{b}_j = \log(X_{ij})$$

Dabei sind:

- $\vec{w}_i, \vec{\tilde{w}}_j$ die Vektoren (Embeddings) der Wörter i und j,
- $b_i, \tilde{b}_j$ Bias-Terme.

4. Gewichtete Kostenfunktion

Um seltene Wortpaare weniger stark zu gewichten, wird eine Gewichtungsfunktion $f(X_{ij})$ verwendet:

$$J = \sum_{i,j=1}^{U} f(X_{ij}) \left(\vec{w}_i^T \vec{\tilde{w}}_j + b_i + \tilde{b}_j - \log(X_{ij}) \right)^2$$

Typisch ist:

$$f(x) = \begin{cases} (x/x_{\max})^\alpha & \text{wenn } x < x_{\max} \\ 1 & \text{sonst} \end{cases} \quad \text{mit } \alpha = 0.75$$

5. Vorteile von GloVe

- Nutzt globale Kookkurrenzstatistik (anders als Word2Vec).
- Robust gegenüber seltenen Wörtern.
- Gut für Analogieaufgaben geeignet.