Natural Language Processing zur Klassifizierung von Unfallberichten

Servan Grüninger

Abschlusspräsentation Praktikum im Team Analysen & Beratung (Bereich VTS, 22.10.18-15.02.19)

suva

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Ziele des Praktikums

1. Etablieren einer Python-Entwicklungsumgebung zur Entwicklung und zum Gebrauch verschiedener Sprachverarbeitungsverfahren

2. Proof of Concept für die Verwendung von Text Embedding Verfahren für Natural Language Processing (NLP)

3. Entwicklung von statistischen Klassifikations-Modellen basierend auf Text Embedding

Ziele des Praktikums

1. Etablieren einer Python-Entwicklungsumgebung zur Entwicklung und zum Gebrauch verschiedener Sprachverarbeitungsverfahren

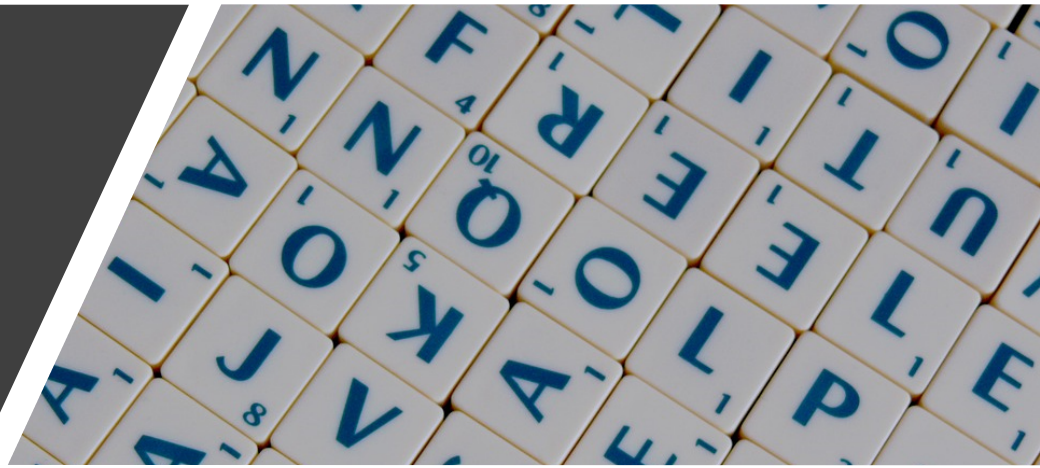
2. **Proof Of Concept für die Verwendung von Text Embedding Verfahren für Natural Language Processing (NLP)**

3. **Entwicklung von statistischen Klassifikations-Modellen basierend auf Text Embedding**

Text Embedding mittels Wortvektoren

Was ist Text Embedding?

- Sammelbegriff für Methoden, um Textinformationen in Zahlen zu verwandeln
- Ermöglicht mathematische und statistische Verarbeitung von Textinformationen
- Mehrere Methoden mit unterschiedlichen Vor- und Nachteilen



Wozu Text Embedding? – Ein Beispiel

Beispielberichte:

Bericht 1: «Der Versicherte stürzte.»

Bericht 2:

Bericht 3:

Umwandlung in Vektoren mit einer Dimension pro verschiedenem Wort («Bag of Words», Dimension d = 3):

	Der	**Versicherte**	**stürzte**
Bericht 1	1	1	1
Bericht 2	-	-	-
Bericht 3	-	-	-

Vektor für Bericht 1: V_1 = [1,1,1]

Vektor für Bericht 2:

Vektor für Bericht 3:

Wozu Text Embedding? – Ein Beispiel

Beispielberichte:

Bericht 1: «Der Versicherte stürzte.»

Bericht 2: «Die Versicherte schnitt sich.»

Bericht 3:

Umwandlung in Vektoren mit einer Dimension pro verschiedenem Wort («Bag of Words», Dimension d = 6):

	Der	**Versicherte**	**stürzte**	**Die**	**schnitt**	**sich**
Bericht 1	1	1	1	0	0	0
Bericht 2	0	1	0	1	1	1
Bericht 3	-	-	-	-	-	-

Vektor für Bericht 1: V_1 = [1,1,1,0,0,0]

Vektor für Bericht 2: V_2 = [0,1,0,1,1,1]

Vektor für Bericht 3:

Wozu Text Embedding? – Ein Beispiel

Beispielberichte:

Bericht 1: «Der Versicherte stürzte.»

Bericht 2: «Die Versicherte schnitt sich.»

Bericht 3: «Die Versicherten stürzten.»

Umwandlung in Vektoren mit einer Dimension pro verschiedenem Wort («Bag of Words», Dimension d = 8)

	Der	**Versicherte**	**stürzte**	**Die**	**schnitt**	**sich**	**Versicherten**	**stürzten**
Bericht 1	1	1	1	0	0	0	0	0
Bericht 2	0	1	0	1	1	1	0	0
Bericht 3	0	0	0	1	0	0	1	1

Vektor für Bericht 1: V_1 = [1,1,1,0,0,0,0,0]

Vektor für Bericht 2: V_2 = [0,1,0,1,1,1,0,0]

Vektor für Bericht 3: V_3 = [0,0,0,1,0,0,1,1]

Die Probleme von «Bag of Words»-Lösungen

1. Dimension der Vektoren für *jeden einzelnen* Bericht steigt an mit der Anzahl unterschiedlicher Wörter *in allen Berichten zusammen*.

2. Informationen zu Wortverwandtschaften und Satzstrukturen gehen komplett verloren.

3. Wörter, die in der Vergangenheit noch nie vorgekommen sind, können nicht verarbeitet werden.

Die Probleme von «Bag of Words»-Lösungen

Text-Vektoren mit einer Dimension pro verschiedenem Wort («Bag of Words»-encoding, Dimension d = 8)

	Der	**Versicherte**	**stürzte**	**Die**	**schnitt**	**sich**	**Versicherten**	**stürzten**
Bericht 1	1	1	1	0	0	0	0	0
Bericht 2	0	1	0	1	1	1	0	0
Bericht 3	0	0	0	1	0	0	1	1

Bericht 2: «Die Versicherte schnitt sich.»

Bericht 4: «Die Versicherte schnitt sich **mit einer Motorsäge**.»

Vektor für Bericht 2: V_2 = [0,1,0,1,1,1,0,0]

Vektor für Bericht 4: V_4 = [0,1,0,1,1,1,0,0]

Besser: Jedem Wort einen Vektor mit fixierter Dimension zuweisen

Beispielberichte:

Bericht 1: «Der Versicherte stürzte.»

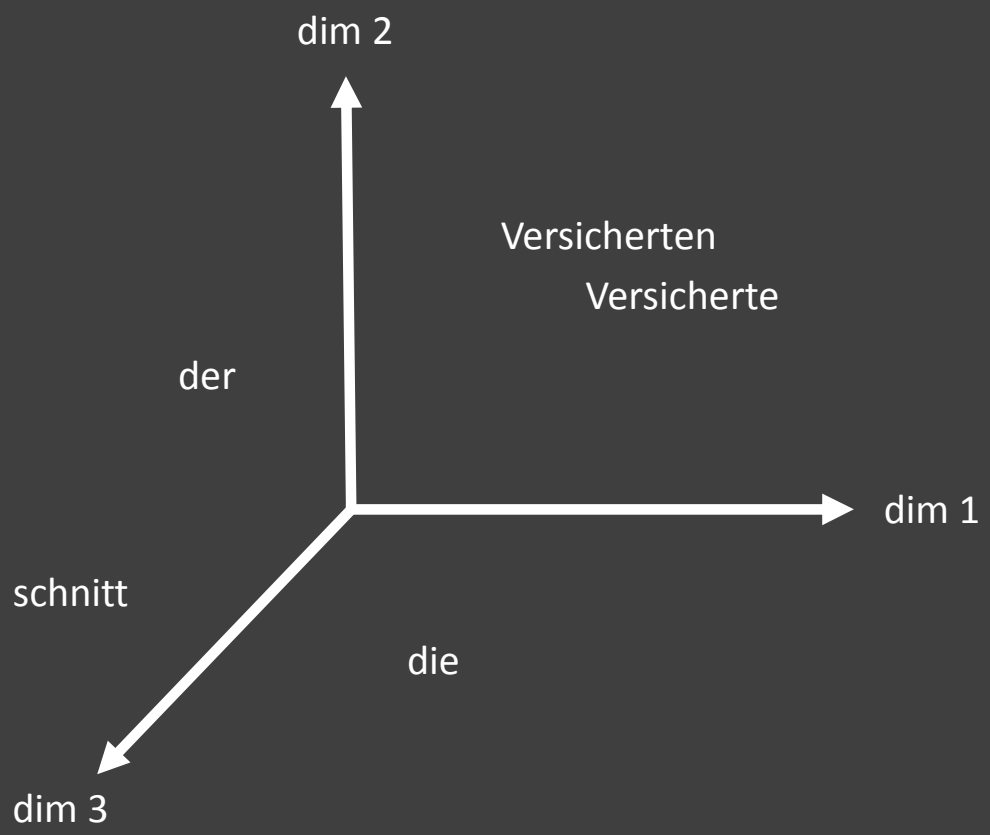
Bericht 2: «Die Versicherte schnitt sich.»

Bericht 3: «Die Versicherten stürzten.»

Umwandlung in Wort-Vektoren mit vordefinierten Anzahl an Dimensionen, z.B. Dimension d = 3.

	dim 1	**dim 2**	**dim 3**
Versicherte	0.23	0.001	0.91
Versicherten	0.21	0.007	0.87
Die	0.89	0.46	-0.21
schnitt	-0.21	0.98	0.03
…	…	…	…

Besser: Jedem Wort einen Vektor mit fixierter Dimension zuweisen



Wortvektoren zu Text-Vektoren kombinieren

Mehrere Möglichkeiten:

- Aufsummieren der Wortvektoren entlang der einzelnen Dimensionen
- Arithmetisches Mittel entlang der einzelnen Dimensionen berechnen
- Mittelwerte der einzelnen Dimensionen minus Principal Components
- …

Hier: Aufsummieren als illustrierendes Beispiel

Wortvektoren zu Text-Vektoren kombinieren

Bericht 1: «Der Versicherte stürzte.»

Umwandlung in Wort-Vektoren

	dim 1	**dim 2**	**dim 3**
Der	0.13	0.01	0.91
Versicherte	0.21	0.007	0.87
stürzte	-0.21	0.98	0.03
Der Versicherte stürzte	?	?	?

Wortvektoren zu Text-Vektoren kombinieren

Bericht 1: «Der Versicherte stürzte.»

Umwandlung in Wort-Vektoren

	dim 1	**dim 2**	**dim 3**
Der	0.13	0.01	0.91
Versicherte	0.21	0.007	0.87
stürzte	-0.21	0.98	0.03
Der Versicherte stürzte	0.13	0.997	1.92

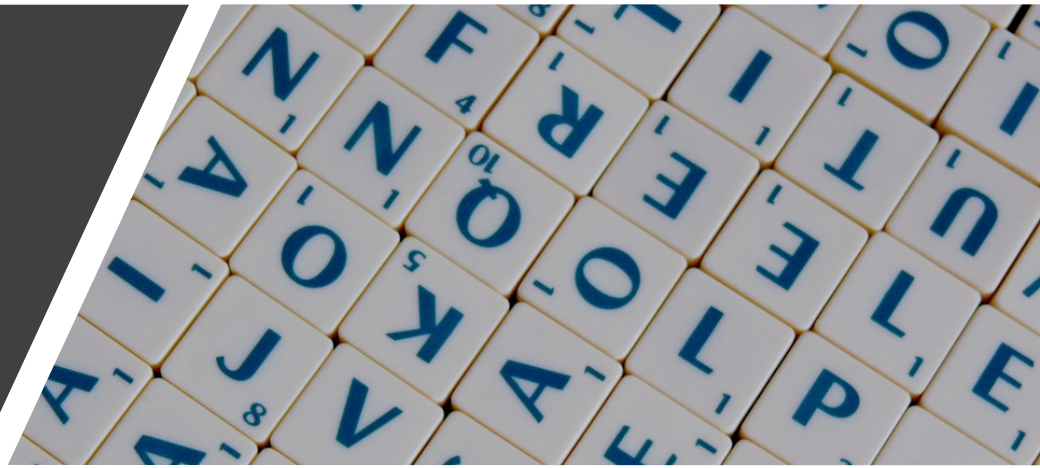
Aufsummieren der Wortvektoren zu Textvektoren:

$$V_{Der} + V_{Versicherte} + V_{stürzte} = \begin{pmatrix} 0.13 \\ 0.01 \\ 0.91 \end{pmatrix} + \begin{pmatrix} 0.21 \\ 0.007 \\ 0.98 \end{pmatrix} + \begin{pmatrix} -0.21 \\ 0.98 \\ 0.03 \end{pmatrix} = \begin{pmatrix} 0.13 \\ 0.997 \\ 1.92 \end{pmatrix} = V_{Bericht1}$$

Wortvektoren erstellen mittels neuronalen Netzen

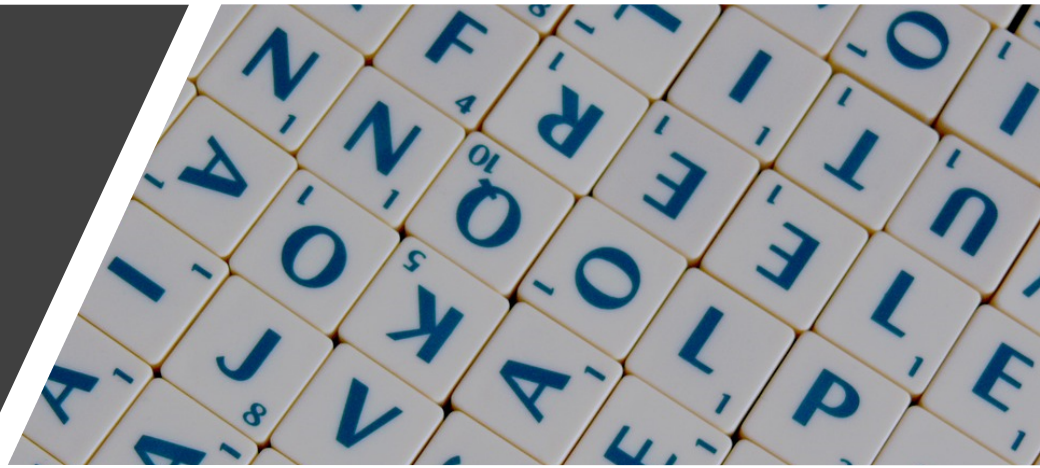
Methoden für effizientes Word-Embedding

- **GloVe** | Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation.
- **Word2Vec** | Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space.
- **FastText (subword information)** | Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information.



Methoden für effizientes Word-Embedding

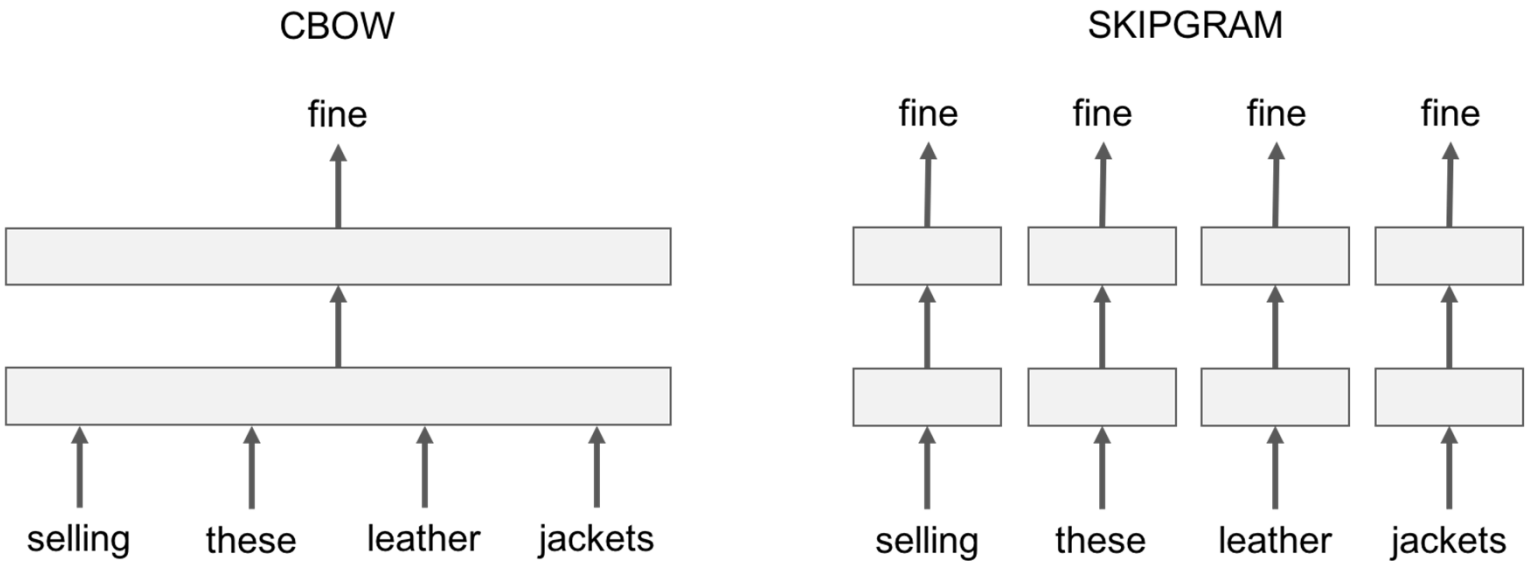
- **GloVe** | Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation.
- **Word2Vec** | Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space.
- **FastText (subword information)** | Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information.



Vorteile von FastText

- Berücksichtigt Informationen aus Satzstruktur und Wortverwandtschaften
- Berücksichtigt Informationen innerhalb von einzelnen Wörtern
- Robuhst gegenubr Schreibfelern
- Kann auch unbekannte Wörter in Wortvektoren umwandeln
- Erkennt semantische Zusammenhänge

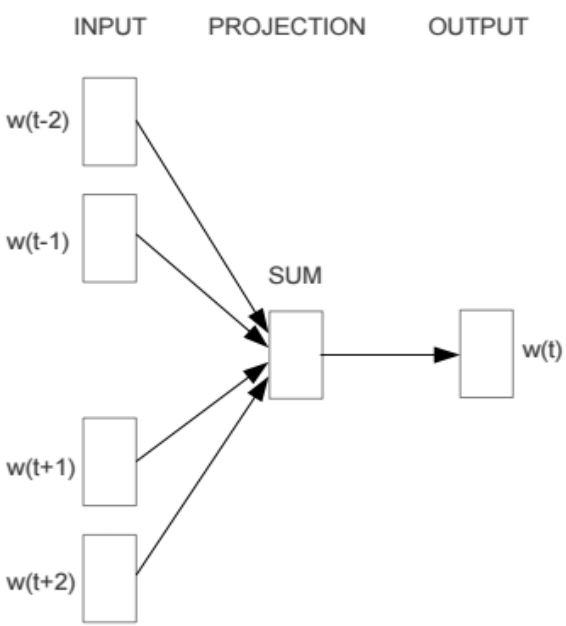
Kontextinformationen erkennen



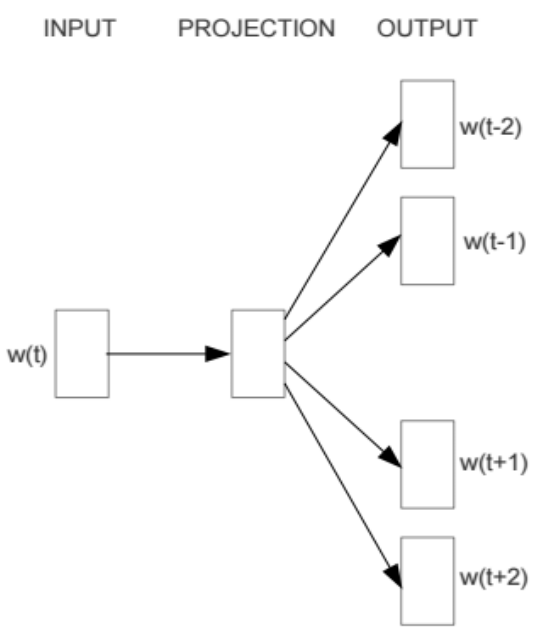
I am selling these fine leather jackets

Bildquelle: fasttext.cc/docs/en/unsupervised-tutorial.html

Kontextinformationen erkennen



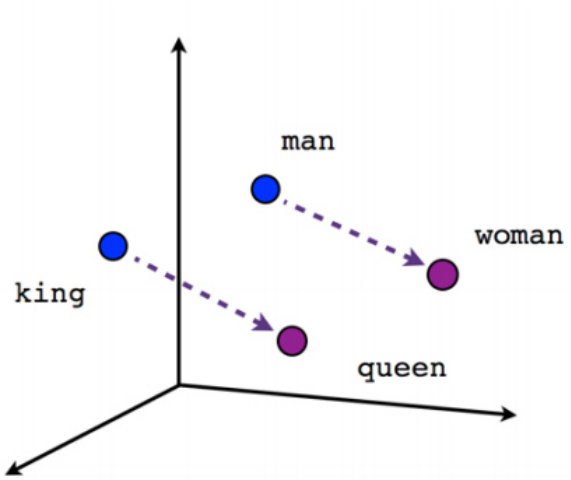
CBOW



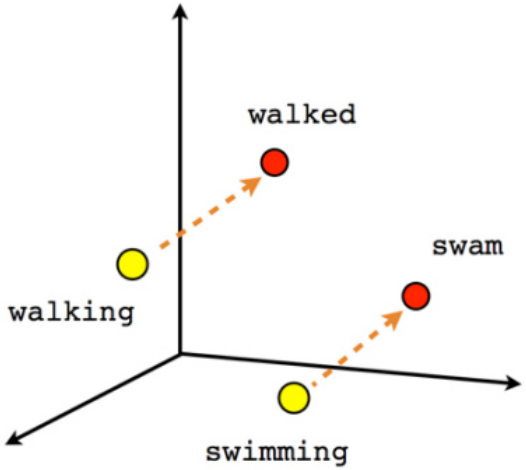
Skip-gram

Bildquelle: hunterheidenreich.com/blog/intro-to-word-embeddings/

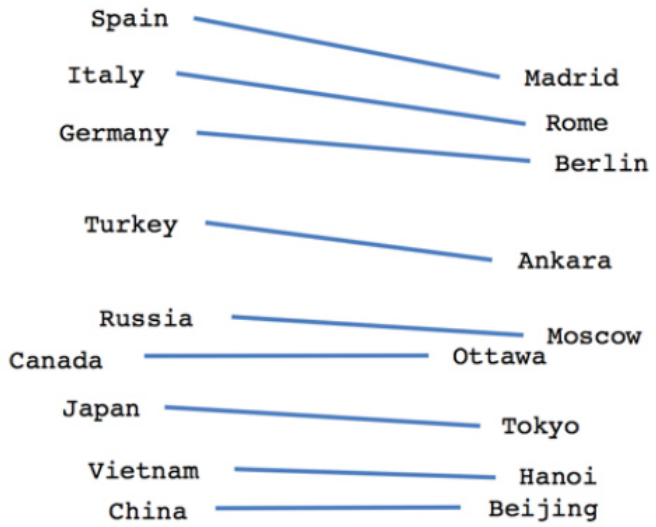
Semantische Informationen abrufen



Male-Female



Verb tense



Country-Capital

Bildquelle: hunterheidenreich.com/blog/intro-to-word-embeddings/

FastText: Subword-Infos als Zusatz (u.a.)

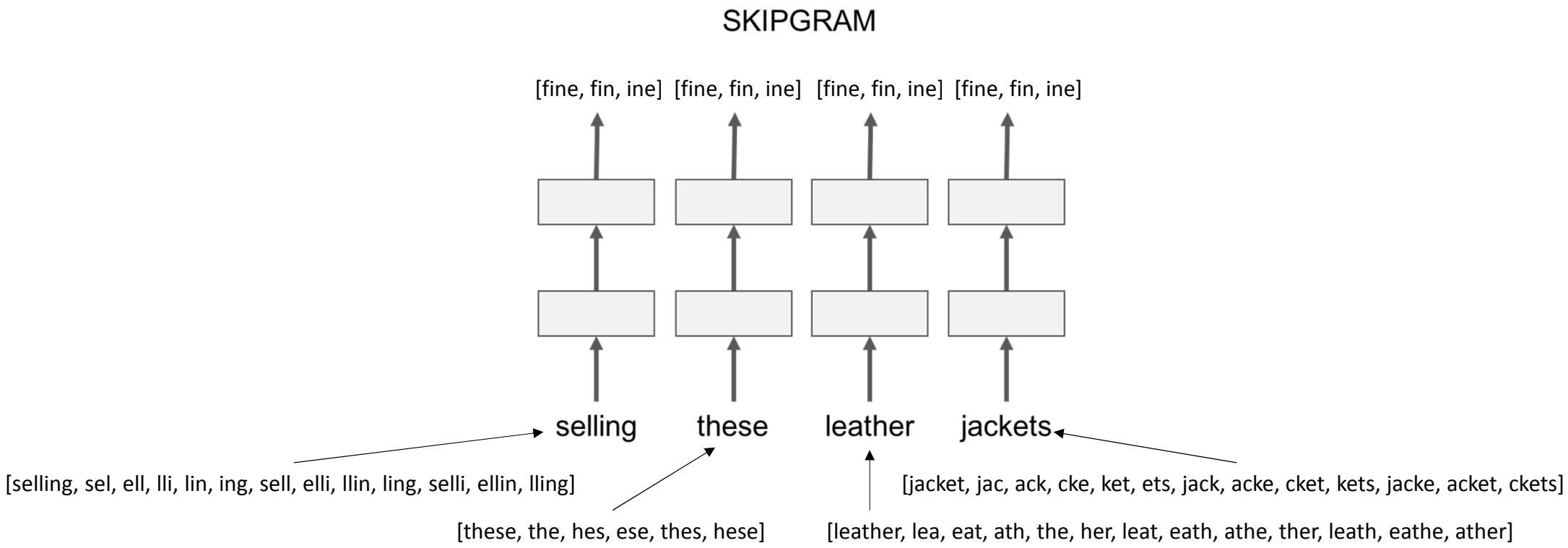
- Idee: Jedes Wort wird als Ganzes wie auch als Sammlung der einzelnen Wortbestandteile (n-grams) repräsentiert.

- Beispiel: Wort «Unfall» & n-gram-Fenster = [3,5]

- «Unfall» wird zu: [Unfall, Unf, nfa, fal, all, Unfa, nfal, fall, Unfal, nfall]
- «Unfälle» wird zu: [Unfälle, Unf, nfä, fäl, äll, Unfä, nfäl, fäll, Unfäl, nfäll, fälle]
- «Unfäle» wird zu: [Unfäle, Unf, nfa, fäl, Unfä, nfäl, fäle, Unfäl, nfäle]

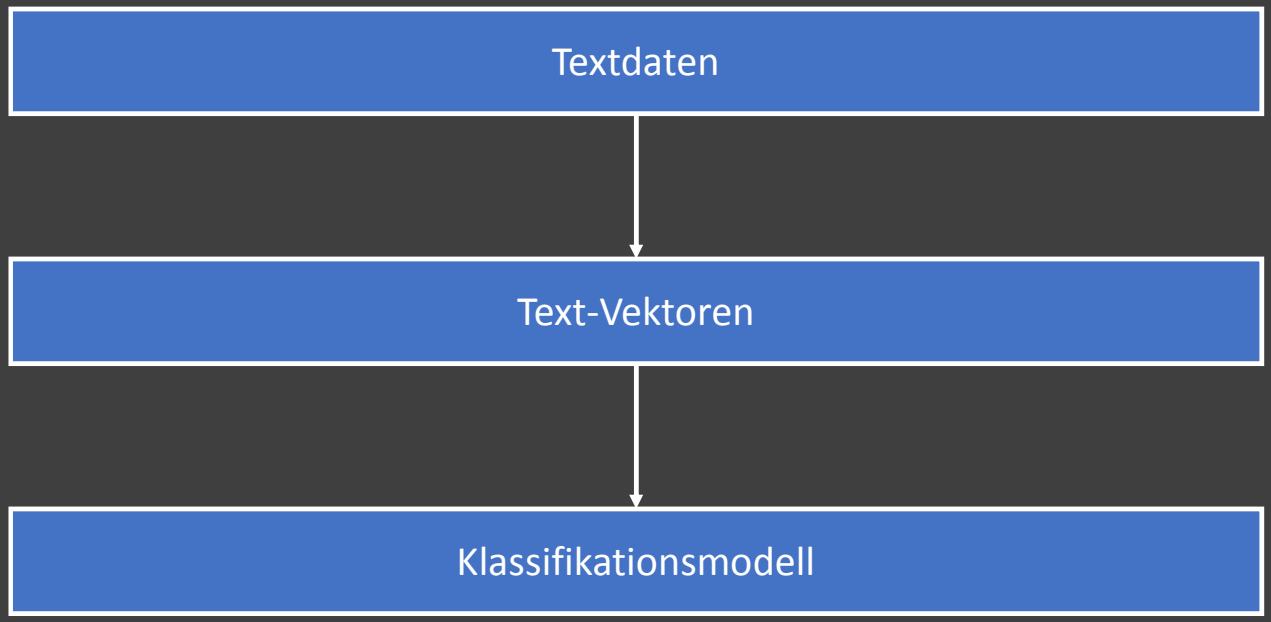
- Erhöht Robustheit gegenüber Schreibfehlern und Flexionen und ist nützlich zum Erkennen von Wortverwandtschaften (z.B. «Bohrer» und «Bohrmaschine»).

Kontext- und Subwordinformation



Wortvektoren nutzen, um Unfallberichte zu klassifizieren

Analyse-Pipeline



Umwandlung der Wortvektoren in Berichtvektoren

- Jeder Bericht besteht aus einer Anzahl *n* von Worten.
- Für jedes dieser Worte erstellen wir einen einzelnen Wortvektor $\boldsymbol{w}_i \in \mathbb{R}^{300}$, wobei $i \in [0,1, \dots, n]$. D.h. wir haben einen Vektor der Dimension 1×300 , wobei jedes Spaltenelement einer reellen Zahl entspricht.
- Um einen Vektor $\boldsymbol{b} \in \mathbb{R}^{300}$ für den ganzen Unfallbericht zu erhalten, berechnen wir einfach den Durchschnitt der einzelnen Wortvektoren des Berichts, d.h. $\boldsymbol{b} = \sum_{i=1}^{n} \boldsymbol{w}_i / n$

Verbesserung der Berichtvektoren

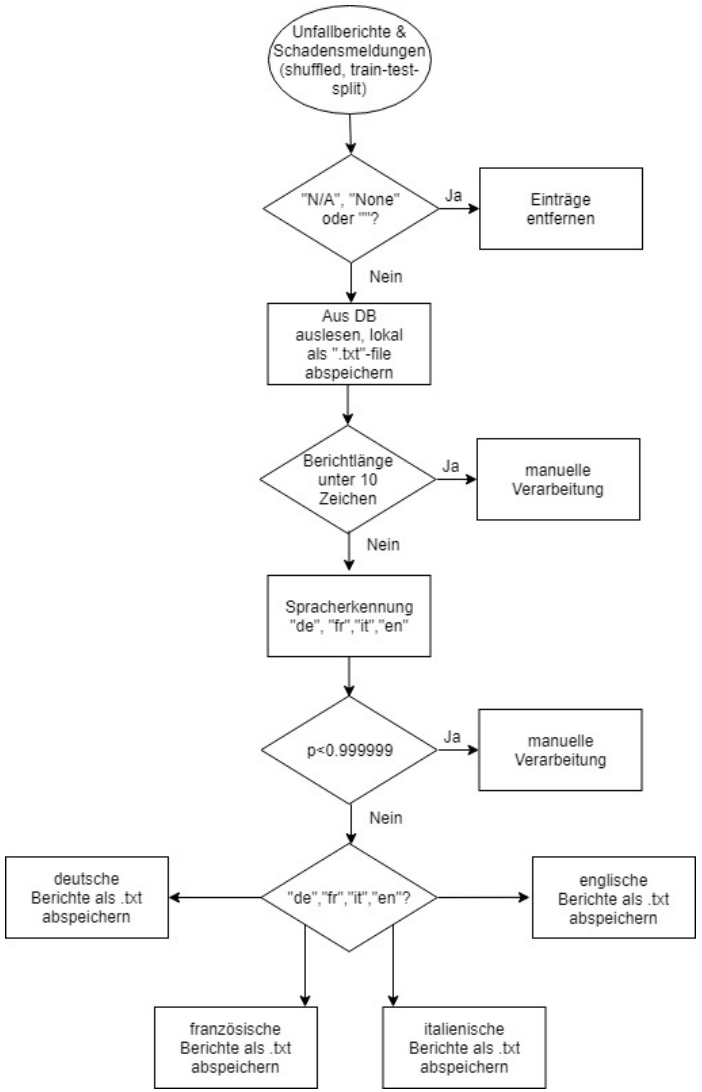
- Alle Berichtvektoren zusammen ergeben die Matrix $B = \begin{bmatrix} \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_m \end{bmatrix}$, wobei *m* die Gesamtzahl der Berichte darstellt.

- Um den Einfluss von Füllwörtern und anderen häufig vorkommenden Wörtern zu reduzieren, ziehen wir erste Hauptkomponente der Matrix von jedem Berichtvektor ab, d.h. $\boldsymbol{b}_i^* = \boldsymbol{b}_i - \boldsymbol{v}_1 \boldsymbol{v}_1^T \boldsymbol{b}_i$

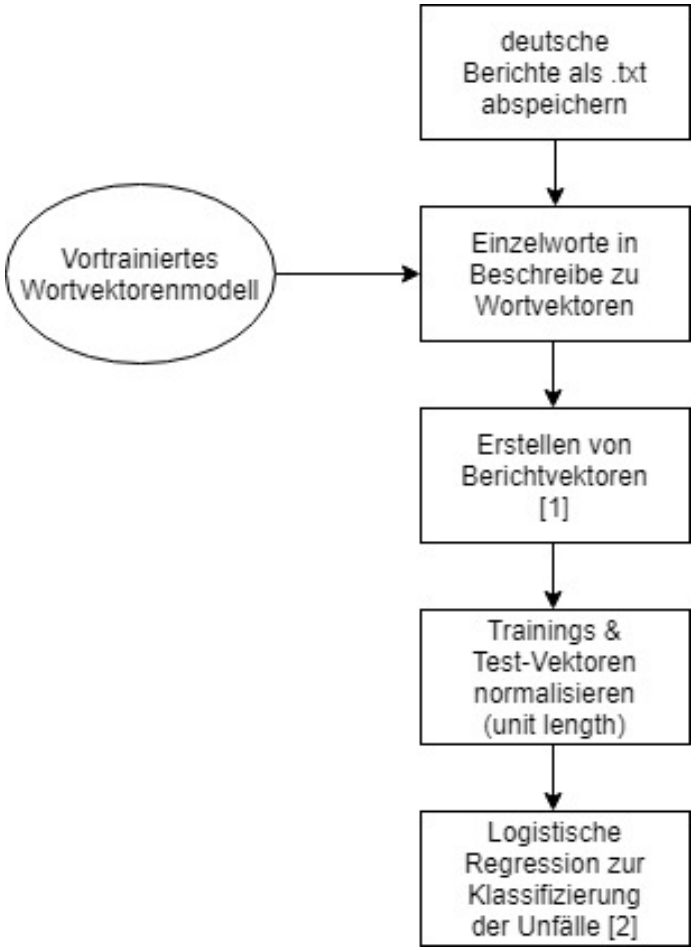
- Hierbei beschreibt $\boldsymbol{v}_1$ die erste Spalte der linken Singulärvektormatrix der Singulärwertzerlegung der Matrix $\boldsymbol{B} = \boldsymbol{U\Sigma V}^T$, d.h. $\boldsymbol{V} = [\boldsymbol{v}_1 \quad \cdots \quad \boldsymbol{v}_{300}]$

Use-Case: Klassifizierung von Elektro-Unfallberichten

Vorbereitung Texte aus DB



Worte zu Wortvektoren



[1] Siehe Folien 27 & 28

[2] Siehe Folie 32

Logistische Regression

Um eine Klassifizierung in Elektrounfälle und restliche Unfälle vorzunehmen, habe ich folgendes Modell angewandt:

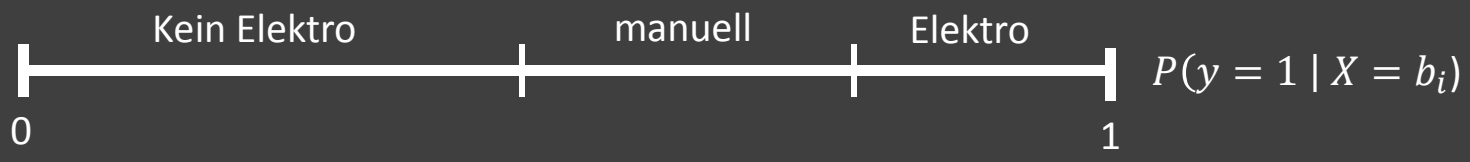
$$P(y = 1 \mid X = b_i) = \frac{\exp(\beta_0 + \boldsymbol{b}_i^T \boldsymbol{\beta})}{1 + \exp(\beta_0 + \boldsymbol{b}_i^T \boldsymbol{\beta})}$$

Hierbei steht $y = 1$ für einen Elektro-Unfall und $y = 0$ für einen Unfall, der kein Elektrounfall ist.

Festlegen der Klassifizierungs-Grenzen

Ohne manuelle Kontrolle: $P(y = 1 \mid X = b_i) > 0.5 \rightarrow$ Elektrounfall

Mit manueller Kontrolle gibt es nicht nur eine, sondern zwei Grenzen:



Grid-Search basierend auf Trainings-Modell mit Trainingsdaten zur Bestimmung optimaler Klassifizierungsgrenzen (Kosten: cost_man = 2, cost_FN = 200, cost_FP = 10)

Vergleich der Resultate – Trainingsset

Confusion Matrix – ohne manuelle Kontrolle

Berichtvektoren & Logistische Regression

DB \ Modell	Pos (E-Unf)	Neg (¬ E-Unf)
Pos (E-Unf)	571	21
Neg (¬ E-Unf)	1358	54926

Balanced Accuracy: 0.97
False Negative Rate: 0.04
Precision: 0.30

EWA

DB \ Modell	Pos (E-Unf)	Neg (¬ E-Unf)
Pos (E-Unf)	567	30
Neg (¬ E-Unf)	363	57538

Balanced Accuracy: 0.97
False Negative Rate: 0.05
Precision: 0.61

Confusion Matrix – inkl. manuelle Kontrolle

Berichtvektoren & Logistische Regression

DB \ Modell	Pos (E-Unf)	manuell	Neg (¬ E-Unf)
Pos (E-Unf)	219	359	14
manuell	-	-	-
Neg (¬ E-Unf)	41	1521	54722

Balanced Accuracy: 0.97
False Negative Rate: 0.06
Precision: 0.84

EWA

DB \ Modell	Pos (E-Unf)	manuell	Neg (¬ E-Unf)
Pos (E-Unf)	459	131	7
manuell	-	-	-
Neg (¬ E-Unf)	50	2012	55839

Balanced Accuracy: 0.99
False Negative Rate: 0.01
Precision: 0.90

Kostenmatrix – inkl. manuelle Kontrolle

Berichtvektoren & Logistische Regression

DB \ Modell	Pos (E-Unf)	manuell	Neg (¬ E-Unf)
Pos (E-Unf)	0	718	2800
manuell	-	-	-
Neg (¬ E-Unf)	410	3042	0

Kosten pro Fall: 0.122
Annahmen:
cost_man = 2, cost_FN = 200, cost_FP = 10

EWA

DB \ Modell	Pos (E-Unf)	manuell	Neg (¬ E-Unf)
Pos (E-Unf)	0	262	1400
manuell	-	-	-
Neg (¬ E-Unf)	500	4024	0

Kosten pro Fall: 0.106
Annahmen:
cost_man = 2, cost_FN = 200, cost_FP = 10

Vergleich der Resultate – Validierungsset

Confusion Matrix – ohne manuelle Kontrolle

Berichtvektoren & Logistische Regression

DB \ Modell	**Pos (E-Unf)**	**Neg (¬ E-Unf)**
Pos (E-Unf)	135	15
Neg (¬ E-Unf)	359	13733

Balanced Accuracy: 0.94
False Negative Rate: 0.10
Precision: 0.27

EWA

DB \ Modell	**Pos (E-Unf)**	**Neg (¬ E-Unf)**
Pos (E-Unf)	120	27
Neg (¬ E-Unf)	97	14396

Balanced Accuracy: 0.90
False Negative Rate: 0.18
Precision: 0.55

Confusion Matrix – inkl. manuelle Kontrolle

Berichtvektoren & Logistische Regression

DB \ Modell	Pos (E-Unf)	manuell	Neg (¬ E-Unf)
Pos (E-Unf)	53	85	12
manuell	-	-	-
Neg (¬ E-Unf)	11	390	13691

Balanced Accuracy: 0.91
False Negative Rate: 0.18
Precision: 0.83

EWA

DB \ Modell	Pos (E-Unf)	manuell	Neg (¬ E-Unf)
Pos (E-Unf)	87	49	11
manuell	-	-	-
Neg (¬ E-Unf)	17	533	13943

Balanced Accuracy: 0.94
False Negative Rate: 0.07
Precision: 0.84

Kostenmatrix – inkl. manuelle Kontrolle

Berichtvektoren & Logistische Regression

DB \ Modell	Pos (E-Unf)	manuell	Neg (¬ E-Unf)
Pos (E-Unf)	0	1700	2400
manuell	-	-	-
Neg (¬ E-Unf)	110	780	0

Kosten pro Fall: 0.24
Annahmen:
cost_man = 2, cost_FN = 200, cost_FP = 10

EWA

DB \ Modell	Pos (E-Unf)	manuell	Neg (¬ E-Unf)
Pos (E-Unf)	0	98	2200
manuell	-	-	-
Neg (¬ E-Unf)	170	1066	0

Kosten pro Fall: 0.24
Annahmen:
cost_man = 2, cost_FN = 200, cost_FP = 10

Beispiele für falsch positive Resultate

Beispiele für «falsch» positive Resultate

Beispiele für falsch negative Resultate

Beispiele für «falsch» negative Resultate

Fazit

Stärken und Schwächen von Wortvektoren

1. Grosse Flexibilität und einfache mathematische Handhabung (z.B. zum Entfernen von statistischem Rauschen aus den Daten)
2. «Natürliche» Kompatibilität mit statistischen Klassifikationsmodellen: Wortvektoren können ohne Anpassungen für Vielzahl von Modellen als Datengrundlage verwendet werden (Logistische Regression, Naive Bayes, Neuronale Netze, Random Forests etc.)
3. Robust in Bezug auf neue Daten, Variationen innerhalb des Textkorpus
4. Semantische Zusammenhänge, Satzstrukturen und Wortverwandtschaften lassen sich abbilden.
5. Zahlreiche vortrainierte Vektor-Modelle existieren bzw. werden laufend weiterentwickelt.
6. Möglichkeit zum Trainieren von eigenen Vektor-Modellen basierend auf Suva-Daten
7. Interpretation der Vektoren ist schwieriger als bei Bag-of-Words-Ansatz