Spezielle Musteranalysesysteme – Projektarbeit
PyTorch Implementierung von 'Combining Language and Vision with a Multimodal Skip-gram Model'

Larissa Strauch
Martrikelnummer: 186513

Dezember 2020

Inhaltsverzeichnis

Kapitel 1

Einleitung

Im klassischen überwachten Lernszenario des maschinellen Lernens wird davon ausgegangen, dass gelabelte Daten für die selbe Aufgabe zur Verfügung stehen. Daraufhin kann ein Modell auf diesem Datensatz trainiert werden, woraufhin von diesem erwartet wird, dass es bei ungesehen Daten der gleichen Aufgabe ähnlich gute Ergebnisse erzielt. Soll nun ein neues Modell trainiert werden, werden wieder gelabelte Daten für diese Aufgabe benötigt. Das traditionelle überwachte Lernparadigma bricht dementsprechend zusammen, wenn nicht genügend gelabelte Daten für eine Aufgabe zur Verfügung stehen, um ein zuverlässiges Modell zu trainieren. Dies bedeutet, dass man nicht auf ein bestehendes Modell zurückgreifen kann, wenn ein Modell für eine neue Aufgabe trainiert werden soll, da sich die Label zwischen den Aufgaben unterscheiden. *Transfer-Lernen* geht darauf nun ein, insofern es ermöglicht mit diesen Szenarien umzugehen, indem bereits vorhandene gelabelte Daten einer verwandten Aufgabe genutzt werden. Es wird versucht dieses Wissen, das bei der Lösung einer Ausgangsaufgabe gewonnen wurde, in der Ausgangsdomäne zu speichern und diese auf das Problem von Interesse anzuwenden.
Der natürlichste Weg für Menschen ist es, Informationen aus verschiedenen Quellen zu extrahieren und zu analysieren. Dies entspricht der Theorie der Semiotik – der Untersuchung der Beziehungen zwischen Zeichen und ihren Bedeutungen auf verschiedenen Ebenen. Die Semiotik untersucht die Beziehung zwischen Zeichen und Bedeutung, die formalen Beziehungen zwischen Zeichen und die Art und Weise, wie Menschen Zeichen in Abhängigkeit vom Kontext interpretieren.

Betrachtet man rein visuelle Zeichen, so führt dies zum Schluss, dass die Semiotik auch durch *Computer Vision* angegangen werden kann, indem interessante Zeichen für die natürliche Sprachverarbeitung extrahiert werden, um die entsprechenden Bedeutungen zu realisieren. In Bezug auf diese Projektarbeit findet der Ansatz zur Vereinigung von *Natural Language Processing* und *Computer Vision* Anklang in den *Distributional Semantics* Modellen. Die Idee hinter diesen besteht darin, dass Wörter in ähnlichen Kontexten eine ähnlichen Bedeutung besitzen sollten. *Distributional Semantics* Modelle werden angewandt, um gemeinsame Semantik zu modellieren, die sowohl auf visuellen Merkmalen als auch auf textuellen wie Wörten basiert. Hierbei besteht die gemeinsame Pipeline darin visuelle Daten auf Wörter abzubilden und darauf vorteilsbezogene Semantikmodelle anzuwenden.
Im Verlauf dieser Projektarbeit wird nun ein Ansatz des *Transfer-Lernens* beschrieben, welcher *Natural Language Processing* und *Computer Vision* miteinander verbindet. Hierbei wird im ersten Kapitel erst einmal auf die Grundlagen des Skip-Gram Modells und der hier implementierten Abhandlung 'Combining Language and Vision with a Multimodal Skip-gram Model' [5] eingegangen, woraufhin im nächsten Kapitel die Implementierung besprochen wird. Abschließend wird noch ein Vergleich zwischen dem Ausgangsmodell und dem multimodalen geschlossen.

Kapitel 2

Grundlagen

Worteinbettungen basieren auf dem Ansatz der Verteilungshypothese von Harris aus dem Jahr 1951, die besagt, dass Wörter, die in denselben Kontexten vorkommen, dazu neigen, ähnliche Bedeutungen zu haben [2]. Eine Worteinbettung liefert einen Wortvektor für jedes Wort. Dies geschieht durch das Extrahieren von Merkmalen aus diesem Wort innerhalb des Kontextes in dem es vorkommt und als Zeichen dafür, dass es einen Platz im Vektorraum einnimmt. Zwei ähnliche Wörter nehmen in diesem Vektorraum nahe beieinander liegende Positionen ein, während Wörter, die sich unterscheiden, viel weiter voneinander entfernt liegen. Auf diese Weise ist es möglich, Entfernungen durch Berechnung der Kosinusdistanz durchzuführen. Es gibt verschiedene Modelle zum Lernen von Wortvektoren. Unter anderem fastText [1], GloVe [8] und word2vec [10]. Hierbei verwendet Word2Vec die beiden Lernmodelle CBOW- und Skip-Gram, wobei auf letzteres im weiteren Verlauf dieser Projektarbeit weiter eingegangen wird.

2.1 Skip-Gram-Modell

Ausgehend von einem Textkorpus versucht die Skip-Gram-Architektur Wortdarstellungen zu entwickeln, welche ausgehend von einem Zielwort die Kontextwörter, welche dieses Wort umgeben, vorhersagt. Das Netzwerk sagt für jedes Wort im Vokabular die Wahrscheinlichkeit vorher, dass es das nahegelegene Wort ist, welches ausgewählt wurde. Die Ausgabewahrscheinlichkeiten werden sich darauf beziehen, wie wahrscheinlich es ist, jedes Vokabularwort in der Nähe des Eingabewortes zu finden.

Wenn beispielsweise dem trainierten Netzwerk das Eingabewort 'Kleid' gegeben wurde, werden die Ausgabewahrscheinlichkeiten für Wörter wie 'Frau' und 'Mensch' viel höher sein als für nicht verwandte Wörter wie 'Fisch' und 'Lampe'.
Mathematisch ausgedrückt erhält das Modell als Eingabe einen One-Hot-Codierungsvektor. Hierbei können die Gewichte der Eingabe- und Ausgabeschicht als $V \cdot N$ Matrix *W* dargestellt werden, wobei jede Zeile von *W* die *N*-dimensionale Wortdarstellung w_v des zugehörigen Wortes der Eingabeschicht ist. Die versteckte Schicht *h* wird durch Multiplikation des One-Hot-Codierungsvektors des Eingabewortes w_I mit der Gewichtsmatrix *W* berechnet [10].

$$h = W^T_{(k,\cdot)} := v^T_{w_I} \tag{2.1}$$

Auf der Ausgabeschicht werden daraufhin *C* Multionimialverteilungen ausgegeben, wobei jede Ausgabe mit der gleichen versteckten Matrix- zu Ausgabematrix berechnet wird [10]:

$$p(w_{c,j} = w_{O,c}|w_I) = y_{c,j} = \frac{\exp\{u_{c_j}\}}{\sum_{j'=1}^{V} \exp\{u_{j'}\}} \tag{2.2}$$

Hierbei ist $w_{c,j}$ das j-te Wort auf dem c-ten Panel der Ausgabeschicht, $w_{O,c}$ ist das tatsächliche c-te Wort in den Ausgabekontextwörtern, w_I ist das einzige Eingabewort, $y_{c,j}$ ist die Ausgabe der j-ten Einheit auf dem c-ten Panel der Ausgabeschicht und $u_{c,j}$ ist die Eingabe der j-ten Einheit auf dem c-ten Panel der Ausgabeschicht [10]. Hierbei maximiert es die Zielfunktion

$$\frac{1}{T}\sum_{t=1}^{T}(\sum_{-c\leq j\leq c, j\neq 0} \log p(w_{c,j}|w_I)) \tag{2.3}$$

, wobei $w_1, w_2, ..., w_I$ die Wörter im Trainingskorpus sind, c die Größe des Fenster um das Zielwort w_I, welches die Menge der durch die induzierten Repräsentationen von w_I vorauszusagenden Kontextwörtern bestimmt.

2.2 Multimodale Skip-Gram-Architektur

In jener Multimodalen Skip-Gram-Architektur werden für eine Teilmenge von Zielwörtern, die Korpustexte von einer visuellen Repräsentation begleitet, welche gleichwohl als Vektor codiert ist und multimodal gemacht wird, indem die Ursprungsformel des Skip-Gram-Modells ausgebaut wird [5].

$$\frac{1}{T}\sum_{t=1}^{T}(L_{ling}(w_t) + L_{vision}(w_t)) \tag{2.4}$$

Hierbei existieren für das Modell zwei verschiedene Versionen – Skip Gram A und Skip Gram B, wobei ersteres in dieser Projektarbeit implementiert wurde.

2.2.1 Skip Gram A

In diesem Modell sollen die Ähnlichkeiten zwischen den sprachlichen und den visuellen Darstellungen direkt erhöht werden. Zu diesem Zweck ist der visuelle Vektor fixiert. Dies bedeutet, dass die Dimensionen des sprachlichen Vektors mit denen des Visuellen in Einklang gebracht werden, was dazu führt, dass die sprachliche Darstellung eines Begriffs näher an seine visuelle Darstellung gebracht wird [5]. Um dies zu erreichen definieren die Autoren die Loss-Funktion

$$- \sum_{w' \; Pn(w)} max(0, \gamma - \cos(u_{w_t}, v_{w_t}) + cos(u_{w_t}, v'_w)) \tag{2.5}$$

, wobei γ die Marge darstellt, u_{w_t} die angestrebte multimodal verbesserte Wortdarstellung die gelernt werden soll, v_{w_t} den korrespondierenden Vektor und v'_w die visuelle Darstellung von Wörtern im Bildwörterbuch, die zufällig aus dern Verteilung $P_n(w_t)$ ausgewählt werden. Hierbei fungieren die visuellen Zufallsdarstellungen als 'negative' Proben, die dazu ermutigen, der eigenen visuellen Darstellung ähnlicher zu sein als der von anderen Wörtern [5].

2.2.2 Skip Gram B

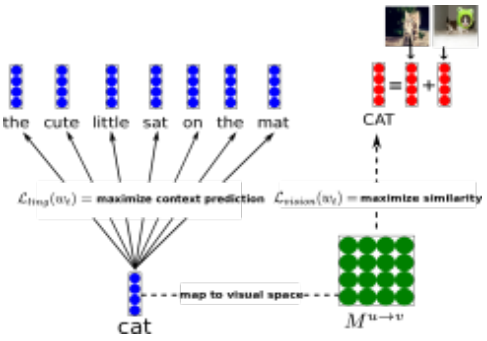

Abbildung 2.1: Skip Gram B

Skip Gram A hat den Nachteil, dass es einen umgehenden Vergleich von sprachlichen und visuellen Darstellungen voraussetzt, was dafür führt, dass ebendiese gleich groß sein müssen.

Skip Gram B hebt dies auf, indem es eine zusätzliche Schicht einfügt, die nebst sprachlichen und visuellen Darstellungen vermittelt. Diese Schicht ist die Schätzung einer cross-modalen Mapping-Matrix von linguistischen und visuellen Repräsentationen, die gemeinsam linguistische Worteinbettungen induziert[5].
Erreicht wird dies, indem in Gleichung 2.5 die Wortrepräsentation u_{wt} durch $z_{wt} = M^{u \to v} u_{wt}$ ersetzt wird, wobei $M^{u \to v} u_{wt}$ die zu induzierende cross-modale Mapping-Matrix darstellt (siehe Abbildung 2.1). Desweiteren wird der Ausgangsgleichung ein L2-Regulierungsterm hinzugefügt um Überanpassungen zu vermeiden [5].

Kapitel 3

Experimenteller Aufbau

Als Grundlage für das hier implementierte Projekt wurde das Skip-Gram-Modell 'Skip-Gram-Model-PyTorch'[1] von *n0obcoder* verwendet. Hierbei wurde die Klassen 'dataset', und 'model' abgeändert, sowie die Klasse 'imageDataset' ergänzt.

3.1 ImageDataset

Die Klasse *imageDataset* ist dafür zuständig Bilder runterzuladen, sie zu transformieren, die Bilder auf die Wörter abzubilden und ein Modell zu trainieren. Hierbei verwendet die Klasse den *CIFAR-100* Datensatz [4], welcher 100 Klassen mit je 600 Bilder enthält. Jede dieser Klassen besteht hierbei aus 500 Trainings- und 100 Testbildern und ist selbst wieder in 20 Oberklassen gruppiert. Jedes Bild kommt mit einem Label zur Klasse und einem Label der Oberklasse zu dem es gehört. Verwendet wird der *CIFAR-100* Datensatz durch Torchvision [6], in welchen er integriert ist. Um die Daten hierbei parallel zu laden wird der 'torch.utils.data.DataLoader' der Bibliothek PyTorch [7] verwendet. Um die geladenen Bilder im Anschluss zu trainieren, implementiert die Klasse eine Methode um *Transfer-Lernen* auf einem vortrainierten ResNet34 [3] anzuwenden. Hierbei wird das gesamte Netzwerk, bis auf die zweite Schicht, also jene mit 128 Merkmalen, eingefroren. Es wurde eine Lernrate von 0.001 gewählt und das Modell in 15 Epochen trainiert.

[1] https://github.com/n0obcoder/Skip-Gram-Model-PyTorch

3.2 Dataset

Im ersten Schritt wird in der *dataset* Klasse ein Korpus der *gensim* [9] Bibliothek verwendet, auf welchem eine Reihe von Vorverarbeitungsschritten durchgeführt werden, durch welche das Vokabular gebaut wird. Hierbei werden im ersten Schritt die Wörter tokenisiert, lemmatisiert und Stoppwörter entfernt, woraufhin im nächsten Schritt über die Token im Korpus iteriert wird und eine Liste von einzigartigen Wort-Token generiert wird. Aufgrunddessen werden zwei Verzeichnisse erstellt, die zwischen den Wörtern und deren Index abbilden und deren Häufigkeit zählt, sowie hinzufügt. Darauf aufbauend, erstellt die Methode *gather_training_data* Zielwort- und Kontextwort-Paare. In dieser Methode werden die ersten Abänderungen zum originalen Skip-Gram-Modell durchgeführt. Um später mit den Daten in einem DataLoader arbeiten zu können, müssen bei der Erstellung der Index-Paare hier schon die Bildvektoren mit eingebracht werden. Hierfür wird als Ausgang ein 0-Wert-Bildtensor der Größe 128 erstellt, sowie negative Proben aus den Bildvektoren gezogen. Wie in der Abhandlung geschrieben, beträgt L_{vision} im Falle, dass kein Bild zum Ziel-Wort vorhanden ist, 0. Um dies in der Methode direkt zu berücksichtigen, wird dort direkt geschaut, ob ein Bild zum Wort existiert. Falls ja, wird der bereits initialisierte Bildtensor durch den passenden Tensor aktualisiert. Im Anschluss, wird der Indexpaararray, welcher vorher nur aus Ziel- und Kontextwort bestand, erweitert durch den Bildtensor, sowie die negativen Proben.

Listing 3.1: Ausschnitt aus der Methode gather_training_data

```
1     context_word_idx = indices[context_word_pos]
2     img = torch.zeros(128) #groesse der Bildvektoren
3     samples = self.create_negative_vision(images, k)
4     img_idx = word_2_img[ix_to_word[indices[center_word_pos]]]
5     if img_idx is not None:
6         img = images[img_idx]
7     training_data_center = torch.tensor(indices[center_word_pos], dtype=torch.long)
8     training_data_context = torch.tensor(context_word_idx, dtype=torch.long)
9     idx_pairs.append((Variable(training_data_center), Variable(training_data_context),
          Variable(img), samples))
```

3.3 Model

In der *model* Klasse wird nun das eigentliche Modell trainiert. Hierbei ist die Verlust-Funktion des erweiterten Skip-Gram-Modells ausschlaggebend. Um diese zu berechnen, wurde die Klasse um die Methode *visual* ergänzt, welche die Formel 2.5 implementiert. Hierbei erhält sie als Parameter das Eingabewort, den zugehörigen Bildtensor, sowie die Proben. Im ersten Schritt berechnet sie die Kosinusähnlichkeit des Eingabewort- und des Bildtensors, woraufhin sie durch alle Bildtensoren der Proben durchläuft und in jedem Schritt einen zweiten Kosinus aktualisiert, indem sie die Kosinusähnlichkeit der Proben- und des Worttensors berechnet. Im Anschluss wird der zweite Kosinus vom Ersten abgezogen und dieser summiert, was als negativer Verlust von der Funktion zurückgegeben wird. Aufgrunderer wird der Verlust der Forward-Funktion im eigentlichen Skip-Gram-Modell ergänzt, indem dieser mit der Verlust-Vision-Funktion addiert wird.

Listing 3.2: Visual-Methode

```
1   def visual(self, input_word, img, samples):
2       cos = nn.CosineSimilarity()
3       tmpLoss = 0
4       loss = 0
5       if img.sum().data != 0:
6           cos_pos = cos(input_word, img)
7           for v_n in samples:
8               cos_neg = cos(input_word, v_n)
9               tmpLoss = tmpLoss + cos_neg
10          max_loss = cos_pos - cos_neg
11          loss = torch.sum(max_loss)
12      return -loss
```

Im Anschluss wird dieses Modell in der main-Funktionen in 10 Epochen, sowie einer Lernrate von 0.001 und einer Batch-Größe von 10 trainiert.

Kapitel 4

Evaluation

Um zu evaluieren wie das Modell abschneidet, wurde in diesem Teil ein Vergleich zwischen dem multimodalen und dem normalen Skip-Gram Modell, sowie einem vortrainierten word2vec Modell der gensim Bibilothek vollzogen. Zu sehen sind die Ergebnisse in Tabelle 4.1. Hierbei ist auffällig, dass sich alle 3 Modelle in nur sehr wenigen Wörtern ähneln. Intuitiv würde ein Mensch behaupten, dass das gensim Modell deutlich besser abschneidet, als die beiden Skip-Gram Modelle. So assoziiert dieses beispielsweise Tiere mit anderen Tieren, wohingegen in den Skip-Gram Modellen das Wort *horse* mit Wörten wie *come* (Skip-Gram A) und *attack* (Skip-Gram) assoziiert. Vergleicht man nur die beiden Skip-Gram Modelle sticht intuitiv kein Modell heraus, das besser als das Andere ist. Anhand der Testwörter sind in beiden Modellen immer mal wieder 'gute' ähnliche Wörter aufgeführt, allerdings ist das Wort *football* in beiden Fällen das einzige Wort, das intuitiv durchgehend gute ähnliche Wörter erzielt.
Alles in allem ist hierbei zu sagen, dass das gewählte Ausgangsmodell nicht die besten Ergebnisse erzielt und daher das multimodale Skip-Gram Modell die Ergebnisse vermutlich nicht steigern konnte.

Eingabewort	Ähnlichstes Wörter Skip-Gram A	Euklidische Distanz	Ähnlichstes Wörter Normales Skip-Gram	Euklidische Distanz	Gensim
horse	come	4.450529	attack	4.893628	dog
	old	4.4828696	later	4.9953647	duck
	later	4.522162	kill	5.158514	dogs
	time	4.550632	life	5.161605	elephant
	young	4.5705028	make	5.1651926	donkey
dog	line	4.968086	live	4.858092	cat
	several	5.0282836	name	4.8626013	dogs
	home	5.034833	continue	4.9091954	horse
	call	5.0529914	day	4.917417	monkey
	far	5.0585866	land	4.919391	pig
man	appear	4.1848764	life	2.546757	was
	woman	4.1862817	later	2.7123027	i
	may	4.2141714	give	2.772801	he
	view	4.242255	go	2.8004305	bad
	time	4.252536	time	2.8036718	even
woman	man	4.1862817	life	3.9965823	child
	boy	4.223959	upon	4.1658316	mother
	television	4.4477863	act	4.185489	whose
	apple	4.5008793	go	4.193925	called
	whose	4.5578322	one	4.2099466	person
football	team	3.0828285	team	3.308419	soccer
	league	3.599337	american	3.6992884	basketball
	play	3.924109	professional	4.0325055	baseball
	american	4.1049414	national	4.0468845	winning
	national	4.1415586	play	4.0539556	sports
apple	woman	4.5008793	computer	3.4413671	windows
	introduce	4.8625016	new	4.199038	microsoft
	television	4.9009356	base	4.2029653	google
	later	5.179545	popular	4.2331033	galaxy
	series	5.186344	time	4.2377996	flash

Tabelle 4.1: Testergebnisse von Skip-Gram-A, dem normalen Skip-Gram Modell und des gensim word2vec Modells

Literaturverzeichnis

[1] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016.

[2] Zellig S. Harris. Distributional structure. *¡i¿WORD¡/i¿*, 10(2-3):146–162, 1954.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[4] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[5] Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. Combining language and vision with a multimodal skip-gram model. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 153–163, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[6] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, page 1485–1488, New York, NY, USA, 2010. Association for Computing Machinery.

[7] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in*

Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc., 2019.

[8] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[9] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA.

[10] Xin Rong. word2vec parameter learning explained. *CoRR*, abs/1411.2738, 2014.