University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Martin Tutek

EXTENDING THE RECURRENT NEURAL NETWORK MODEL FOR IMPROVED COMPOSITIONAL MODELLING OF TEXT SEQUENCES

DOCTORAL THESIS

Zagreb, 2022

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Martin Tutek

EXTENDING THE RECURRENT NEURAL NETWORK MODEL FOR IMPROVED COMPOSITIONAL MODELLING OF TEXT SEQUENCES

DOCTORAL THESIS

Supervisor: Professor Jan Šnajder, PhD

Zagreb, 2022

Sveučilište u Zagrebu

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Martin Tutek

PROŠIRENJE MODELA POVRATNIH NEURONSKIH MREŽA ZA POBOLJŠANO KOMPOZICIJSKO MODELIRANJE TEKSTNIH SLJEDOVA

DOKTORSKI RAD

Mentor: prof. dr. sc. Jan Šnajder

Zagreb, 2022.

The doctoral thesis has been made at the University of Zagreb, Faculty of Electrical Engineering and Computing, at the department of electronics, microelectronics, computer and intelligent systems, as part of the Text analysis and knowledge engineering laboratory (TakeLab).

Supervisor: Professor Jan Šnajder, PhD

The doctoral thesis consists of: 93 pages

Doctoral thesis num.:

About the Supervisor

Jan Šnajder has received his BSc, MSc, and PhD degrees in Computer Science from the University of Zagreb, Faculty of Electrical Engineering and Computing (FER), Zagreb, Croatia, in 2002, 2006, and 2010, respectively. From September 2002 he was working as a research assistant, from 2011 as Assistant Professor, from 2016 as Associate Professor, and from 2021 as Full Professor at the Department of Electronics, Microelectronics, Computer and Intelligent Systems at FER. He was a visiting researcher at the Institute for Computational Linguistics at the University of Heidelberg, the Institute for Natural Language Processing at the University of Stuttgart, the National Institute of Information and Communications Technology in Kyoto, and the University of Melbourne. He participated in a number of research and industry projects in the field of natural language processing and machine learning. He is the principal investigator on a HRZZ installation grant project and a HAMAG-BICRO proof-of-concept project, and a researcher on a UKF project. He has (co-) authored more than 120 papers in journals and conferences in natural language processing and information retrieval, and has been reviewing for major journals and conferences in the field. He is the lecturer in charge for six courses at FER and has supervised and co-supervised more than 100 BA and MA theses. He is a member of IEEE, ACM, ACL, the secretary of the Croatian Language Technologies Society, the co-founder and secretary of the Special Interest Group for Slavic NLP of the Association for Computational Linguistics (ACL SIGSLAV). He is a member of the Centre of Research Excellence for Data Science and Advanced Cooperative Systems and the associate editor of the Journal of Computing and Information Technology. He has been awarded the Silver Plaque "Josip Lončar" in 2010, the Croatian Science Foundation fellowship in 2012, the fellowship of the Japanese Society for the Promotion of Science in 2014, and the Endeavour Fellowship of the Australian Government in 2015.

O mentoru

Jan Šnajder diplomirao je, magistrirao i doktorirao u polju računarstva na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva (FER), 2002., 2006. odnosno 2010. godine. Od 2002. godine radio je kao znanstveni novak, od 2011. godine kao docent, od 2016. godine kao izvanredni profesor, a od 2021. kao redoviti profesor na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave FER-a. Usavršavao se na Institutu za računalnu lingvistiku Sveučilišta u Heidelbergu, Institutu za obradu prirodnog jezika Sveučilišta u Stuttgartu, Nacionalnome institutu za informacijske i komunikacijske tehnologije u Kyotu te Sveučilištu u Melbourneu. Sudjelovao je na nizu znanstvenih i stručnih projekata iz područja obrade prirodnog jezika i strojnog učenja. Voditelj je uspostavnog projekta HRZZ-a i projekta provjere koncepta HAMAG-BICRO-a te je istraživač na projektu UKF-a. Autor je ili suautor više od 120 znanstvenih radova u časopisima i zbornicima međunarodnih konferencija u području obrade prirodnog jezika i pretraživanja informacija te je bio recenzentom za veći broj časopisa i konferencija iz tog područja. Nositelj je šest predmeta na FER-u te je bio mentorom ili sumentorom studentima na više od 100 preddiplomskih i diplomskih radova. Član je stručnih udruga IEEE, ACM, ACL, tajnik Hrvatskoga društva za jezične tehnologije te suosnivač i tajnik posebne interesne skupine za obradu prirodnog jezika za slavenske jezike pri udruzi za računalnu lingvistiku (ACL SIGSLAV). Član je Znanstvenog centra izvrsnosti za znanost o podacima i kooperativne sustave te je pridruženi urednik časopisa Journal of Computing and Information Technology (CIT). Dobitnik je Srebrne plakete "Josip Lončar" 2010. godine, stipendije Hrvatske zaklade za znanost 2012. godine, stipendije Japanskog društva za promicanje znanosti 2014. godine te stipendije australske vlade Endeavour 2015. godine.

“Given the pace of technology, I propose we leave math to the machines and go play outside."
– Calvin; from Calvin and Hobbes by Bill Watterson

I dedicate this thesis to:

my parents, who have unwaveringly supported me every step of the way;
colleagues, who broke long working days into manageable chunks;
friends, who listened to my endless rants and still talk to me;
my supervisor, who was the best I could have hoped for.

Abstract

The thesis focuses on exploring extensions to the recurrent neural network (RNN) algorithm for natural language processing (NLP) in terms of improving its capabilities of semantic composition, investigating the possible benefits of leveraging multi-prototype word representations and improving its overall interpretability. While RNNs have received a strong competitor in form of the Transformer model, both approaches to processing natural language sequences possess their own set of issues.

This research investigates methods of inducing sparsity in neural networks in order to learn shared sense representations, a paradigm that differs from the standard sense-specific representation approaches where even synonymous words contain distinct sense representations. More specifically, we evaluate whether it is possible to learn such a model by virtue of autoencoder style training on unstructured natural language data. The research also tackles the problem of semantic composition in recurrent networks, first analysing and resolving the issues recurrent networks face when propagating information laterally and vertically. Based on these findings, the thesis introduces a novel approach for building recursive representations of language which is better suited to the hierarchical phrasal structure of language. The results along both research avenues – the one of investigating shared sense representations and one investigating semantic compositionality – offer valuable insights into avenues of improving recurrent models, and demonstrate improvements with respect to interpretability of models.

Keywords: word representations, multiprototype representations, semantic composition, interpretability, recurrent neural networks, natural language processing

Proširen sažetak

Proširenje modela povratnih neuronskih mreža za poboljšano kompozicijsko modeliranje tekstnih sljedova

Obrada prirodnog jezika (OPJ) je područje računarske znanosti koje omogućava strojnu obradu i razumijevanje podataka izraženoga tekstom. Kao i cijelo područje umjetne inteligencije, obrada prirodnog jezika je prošla značajnu transformaciju popularizacijom neuronskih pristupa, koji su postavil impresivne rezultate u podzadacima OPJ poput klasifikacije teksta, strojnog prevođenja i zaključivanja temeljem teksta. Svi duboki neuronski modeli dijele niz zajedničkih komponenti, od kojih su nam dvije od posebnog interesa: (1) ulazne sastavnice, poput riječi, se trebaju prikazati kao guste vektorske reprezentacije kako bi ih strojevi mogli razumjeti, (2) semantičku kompozicije individualnih sastavnica u značenje teksta i (3) zbog iznimno velikog broja parametara neuronskih modelima oni inherentno nisu transparentni te je razumijevanje njihovog slijeda odluka otežano. Fokus ovog rada je upravo na pristupima rješavanju navedenih problema – istraživanjem tehnika distribucijske semantike za prikaz riječi vektorima i tehnika regularizacije za poboljšanje transparentnosti dubokih neuronskih modela.

Uz relacijsku semantiku, drugi etablirani pristup predstavljanju značenja riječi je distribucijska semantika, koja promatra povezanost riječi kroz prizmu njihovog supojavljivanja u velikim korpusima teksta. Metode temeljene na distribucijskoj semantici često se koriste za postavljanje inicijalnih vrijednosti gustih prikaza riječi, pri čemu očekujemo da je semantička sličnost gustih prikaza dvaju riječi otprilike jednaka nekoj mjeri sličnosti njihovih vektora. Večina pristupa temeljenih na distribucijskoj semantici pripadaju paradigmi jednoprototipnih prikaza koja svakoj riječi prirodjeljuje jedan gusti vektorski prikaz te sadrži određene nedostatke, specifično pri obradi homonimnih i polisemnih riječi. Alternativni pristupi specijalizirani za obradu homonimnih i polisemnih riječi su pristup višeprototipnih prikaza i dijeljenih prikaza, od kojih nam je potonji od izrazitog interesa zbog poboljšane memorijske složenosti i olakšanog razumijevanja rada modela.

Ovaj rad istražuje poboljšanja pristupa semantičkoj kompoziciji gustih reprezentacija temeljenih na povratnih neuronskim mrežama i algoritmu pozornosti, pristupa učenju dijeljenih gustih prikaza riječi te poboljšanju olakšanog razumijevanja rada neuronskih modela kroz tehnike regularizacije. Središnja hipoteza ovog rada jest da su problemi semantičke kompozicije i prikaza riječi povezani – te da se ciljanim poboljšanjem tehnika koje se bave navedenim problemima može, uz povećanu točnost, ostvariti i poboljšana transparentnost kod dubokih neuronskih modela. Ova hipoteza ispitana je kroz tri eksperimentalna doprinosa, koji ciljaju dati odgovor na tri istraživačka pitanja (IP):

- IP1 – Koliko precizno povratne neuronske mreže prikazuju re čeničnu semantiku i koje su im mane semantičke kompozicije?

- •IP2 – Jesu li višeprototipni ili dijeljeni prikazi rije či prikladna paradigma za neuronske modele te poboljšavaju li takvi prikazi performanse modela na ciljnim zadacima?
- •IP3 – Može li poboljšanje modela semanti čke kompozicije kod povratnih neuronskih mreža istovremeno poboljšati kvalitetu rečeničnih prikaza i transparentnost njihove izgradnje?

1. Uvod

U prvome poglavlju (“*Introduction*") opisani su motivacija i ciljevi doktorskog istraživanja. Najprije se opisuju pristupi predstavljanju značenja riječi kroz guste vektorske prikaze te neuronski modeli semantičkoj kompoziciji značenja individualnih riječi u značenje rečenica ili teksta. Nastavno, izoliraju se nedostaci svakog od prethodno navedenih pristupa te se opisuju dosadašnji pristupi ublaživanju tih nedostataka. Završno, postavljaju se istraživačka pitanja na koja ovaj rad cilja dati odgovore te se predstavlja kratak sažetak sadržaja doktorskog rada..

2. Pozadina

U drugome poglavlju (“*Background*") opisuje se domena istraživanja kroz pregled zadataka obrade prirodnog jezika kojih se doktorsko istraživanje dotiče. Konkretno, opisuju se pristupi vektorskim prikazima riječi, posebice onih specijaliziranih za polisemiju kao i kontekstualizirani pristupi vektorskim prikazima riječi. Potom, daje se pregled tehnika razumijevanja rada neuronskih modela kroz njihove tri kategorije – (1) metoda baziranih na gradijentima, (2) metoda baziranih na unatražnoj propagaciji te (3) metoda baziranih na surogatnim modelima.

3. Poboljšanje vjerodostojnosti objašnjenja algoritmom pozornosti

U trećemu poglavlju (“*Improving Faithfulness of Attention Explanations*") opisuje se provedeno istraživanje tehnika regularizacije kojima je cilj bio poboljšanje vjerodostojnosti objašnjenja ponašanja povratnih neuronskih modela korištenjem algoritma pozornosti. U okviru istraživanja predložene su dvije tehnike regularizacije s ciljem ublažavanja dvaju fundamentalnih problema pri propagaciji informacija u povratnim neuronskim mrežama – (1) horizontalnog curenja informacija i (2) vertikalne blokade informacija. U standardnim pristupima objašnjavaju povratnih neuronskih modela kroz algoritam pozornosti pretpostavlja se da svaki skriveni prikaz dobiven obradom povratne neuronske mreže odgovara upravo kontekstualizaciji prikaza odgovarajuće ulazne riječi. Eksperimentalno je pokazano da prethodno navedena pretpostavka ne vrijedi, te su predložene dvije tehnike regularizacije povratnih modela koje osiguravaju zadržavanje informacija iz ulaznih prikaza u odgovarajućim skrivenim prikazima povratnih modela. Jedna tehnika regularizacije je inspirirana maskiranim modeliranjem jezika (MLM), gdje se dio ulaznih riječi slučajno maskira, te je dodatni zadatak modela rekonstruirati maskirane riječi. Korištenjem navedenog dodatnog zadatka pri učenju modela osigurava se da modeli u skrivenim reprezentacijama sadrže informacije potrebne za rekonstrukciju ulaznih riječi, čime se osnažuje poklapanje između informacija u ulaznim i skrivenim prikazima. Druga tehnika regularizacije je primjena L_2 vezanja koja uvodi eksplicitnu težinsku vezu između ulaznih i skrivenih reprezentacija te kažnjava njihovo odstupanje po L_2 normi.

Vrednovanje modela osnaženih s predstavljenim tehnikama regularizacije provelo se na nizu klasifikacijskih skupova podataka za engleski jezik iz različitih domena poput recenzija filmova, medicine i kategorizacije novinskih članaka. Pri vrednovanju modela pratilo se više kriterija: (1) modeli su trebali ostvariti zadovoljavajuću klasifikacijsku točnost na podatcima te (2) modeli su trebali pokazati zadržavanje informacija iz ulaznih reprezentacijama u skrivenim reprezentacijama. Obje predložene tehnike regularizacije demonstrirale su značajna poboljšanja u zadržavanju ulaznih informacija uz minimalni gubitak klasifikacijske točnosti.

Uz kvantitativnu analizu kroz navedene kriterije provedena je i kvalitativna analiza tehnika regularizacije kroz analizu dodijeljene važnosti pojedinačnim skrivenim reprezentacijama. Pregledom niza primjera iz validacijskog skupa analiziranih skupova podataka pokazano je da primjenjene tehnike regularizacije smanjuju bitnost koju modeli pridodijeljuju skrivenim prikazima riječi koje ne sadrže informacije bitne za analiziran klasifikacijski zadatak.

4. Učenje dijeljenih prikaza riječi

U četvrtome poglavlju ("*Learning Shared Word Representations*") opisuje se implementacija samokodirajućeg rijetkog modela za učenje dijeljenih prikaza riječi. U okviru predloženog modela prvo su predstavljene tehnike koje se koriste za ugrađivanje rijetkosti u duboke neuronske modele, konkretnije *Gumbel-softmax*, čvrsta Kumaswaramy distribucija te kvantizirani varijacijski samokodirajući modeli. Navedene tehnike su nužne za učenje dijeljenih prikaza riječi radi problema redundancije informacija – ukoliko se ne uvede rijetki sloj, neuronski modeli mogu konvergirati u neželjeno rješenje gdje sve ulazne informacije ostaju uspješno očuvane u skrivenom sloju, čime se ne postiže željena generalizacija i dijeljenje prikaza istoznačnica. Osim same tehnike uvođenja rijetkosti u modele, opisuje se i samokodirajući model učen samonadziranim učenjem. Samonadzirano učenje je u zadnje vrijeme standardan pristup predučenju dubokih modela, populariziran radi iscrpne količine tekstnih podataka na internetu koji se mogu iskoristiti za učenje raspodjele supojavljivanja riječi u jeziku. Nadalje, opisana je bitna stavka samokodirajućeg modela – rekonstrukcijska mreža. Pristup rekonstrukciji tekstnih podataka može se napraviti na dva načina: (1) nezavisnom rekonstrukcijom svake riječi s ulaza i (2) zavisnom rekonstrukcijom s lijeva na desno. Dok prvi pristup riskira određenu razinu redundancije radi nezavisnosti dekodiranih riječi, te se često pojavljuju simptomi generativnih modela teksta poput ponavljanja čestih riječi, zavisni pristup ima manu da kroz uvođenje dodatnih parametara možda pretjerano pojednostavljuje zadatak rekonstrukcije – čime se ponovno ne postiže zadovoljavujća razina generalizacije. Konačno, opisuju se metodološke varijante postavljanja početnih vrijednosti dijeljenih prikaza riječi te slučajnog izbacivanja dijeljenih prikaza, oboje korisnih za koznistentno postizanje bolje razine generalizacije.

Vrednovanje samokodirajućih modela za učenje dijeljenih prikaza riječi provedeno je na dva klasifikacijska skupa podataka i jednom skupu podataka za mjerenje sličnosti značenja riječi u kontekstu. Rezultati pokazuju da sve razmatrane varijante samokodirajućih modela ostvaruju razumne performanse na zadatku mjerenja sličnosti značenja riječi u kontekstu, no i dalje zaostaju za ponajboljim modelima. Na klasifikacijskim zadacima rezultati ponajbolje varijante samokodirajućeg modela za učenje dijeljenih prikaza riječi čak ostvaruju najbolji rezultat među razmatranim modelima – čime dobivamo naznake da pristup kvantiziranim varijacijskim samokodirajućim modelom vrijedi dublje razmotriti u budućnosti.

5. Iterativni rekurzivni mehanizam pozornosti

U petome poglavlju ("*The Iterative Recursive Attention Mechanism*") opisuje se predloženo proširenje mehanizma pozornosti za povratne neuronske mreže s ciljem poboljšanja razumijevanja načina na koji neuronski modeli donose odluke. Sam model je opisan kroz tri zasebne faze obrade podataka u unaprijednom prolazu: (1) fazu kodiranja, (2) fazu pozornosti i (3) fazu klasifikacije. Glavni doprinos predloženog modela nalazi se u fazi pozornosti, gdje se klasični mehanizam pozornosti proširio na način da se prikaz dobiven u jednom koraku mehanizma pozornosti dodaje u skup vektorskih prikaza na koje se primjenjuje pozornost u idućem koraku – omogućavajući hijerarhijsko učenje prikaza jezika. Dodatna korist rekurzivne pozornosti je poboljšana interpretabilnost. U situacijama gdje se model susreće sa rečenicama koje sadrže kontrastni sentiment, prethodno je odluka između polarnosti rečenice morala biti donešena u jednom koraju pozornosti, pri čemu bi se često naglasio samo dominantni polaritet. Razdvajanjem obrade u više uzastopnih koraka modelu se daje mogućnost kompartmentalizacije polariteta u odvojene korake, čime svaki korak izolira jedan od kontrastnih aspekata ulazne rečenice.

Kvantitativno vrednovanje predloženog modela s rekurzivnom pozornosti provedeno je na dva klasifikacijska skupa podataka iz domene analize sentimenta na engleskom jeziku, dok je kvalitativno vrednovanje provedeno na slučajno odabranim primjerima iz skupova za validaciju. Kvantitativna evaluacija je u trenutku objave modela ostvarila najbolje rezultate na skupu podataka SST-2 te kompetitivne rezultate na skupu podataka IMDb. Kvalitativnom evaluacijom pokazano je da model koristi rekurzivni algoritam na željen način – obrađujući kontrastne aspekte polarnosti u zasebnim koracima pozornosti, te ih kombinirajući u posljednjem koraku pred klasifikaciju.

6. Rasprava

U šestome poglavlju ("*Discussion*") istraživanje provedeno unutar doktorskog rada se sagledava iz šire perspektive, prvo analizirajući implikacije za dvije glavne varijante neuronskih modela za obradu teksta – povratnih neuronskih mreža i mreža Transformer. Potom se analizira korisnost transparentnosti modela u današnjem okruženju neuronskih modela, s naglaskom na kompromis između performansi na ciljnim zadacima i razumijevanja načina na koji modeli donose odluke. Konačno, razmatra se motivacija za eksplicitnim modeliranjem pojedninačnih smislova riječi – pristupa koji je postao manje bitan nedavnim napretkom u učenju kontekstualiziranih prikaza riječi.

6. Zaključak

Posljednje, šesto poglavlje ("*Conclusion*") ukratko sažima predložene modele i njihove implikacije za buduće pravce istraživanja. Konkretno, podrobnije se analiziraju pravci poboljšanja protoka informacija kroz povratne neuronske mreže, problema koji je dugo vrijeme poznat kroz simptome eksplodirajućih i nestajućih gradijenata. Idući interesantan pravac je učenje dijeljenih prikaza riječi, čija korist nije očita u performansama modela na razmatranim klasifikacijskim zadacima usprkos jasnoj motivaciji iz teorije linvistike. Unutar analize izoliraju se tri glavna pravca za budući rad u okviru učenja dijeljenih prikaza riječi koji su se pokazali obećavajućim Konačno, analizira se pravac učenja rekurzivnih reprezentacija teksta koje poboljšavaju transparentnost neuronskih modela uz dodatnu korist poboljšavanja performansi na razmatranim klasifikacijskim zadacima.

Ključne riječi: značenje riječi; povratne neuronske mreže; algoritam pozornosti; transparentnost dubokih neuronskih mreža; obrada prirodnog jezika

Contents

Chapter 1

Introduction

Natural language processing (NLP) is a subfield of computer science focusing on enabling machines to process and understand natural language data. Along with other areas of artifical intelligence, NLP has been affected by the now not-so-recent neural revolution, which helped propel the state of the art across tasks such as sequence classification, natural language inference, and text generation. All deep neural models have two aspects in common: (1) they require input tokens, such as words, to be mapped to numeric representations, which can then be "understood" by machines, and (2) due to the sheer number of parameters, most of the models are black-box and cannot be understood by human practitioners as their shallow counterparts.

The dominant paradigm used for learning word representations for neural models is *distributional semantics*. Following the *distributional hypothesis* [1, 2], which states that *"a word is characterized by the company it keeps"*, distributional semantics relies on semantically related words being highly likely to occur in similar contexts. For example, we can imagine a number of words that can complete the following sentence:

"He took a _____ back home."

such as *"cab"*, *"bus"*, or *"tram"* – all of which share in common being the means of transportation. Distributional semantics leverages this fact and, through constructing supervised learning tasks from large amounts of unstructured text available in the era of World Wide Web, induces semantic representations of words from instances of contexts the word appears in.

Vector space models for word meaning [3] represent each word by a dense vector representation, the idea being that semantically similar words are embedded as points that are close to each other in the vector space. We provide a visualization of the various approaches to word representations in Figure 1.1. While initial approaches to learning word representations leveraged coocurrence matrix decomposition, neural predictive approaches [4] emerged as the predominant method. Nevertheless, an important question still lingered: should each word be assigned a single word embedding (Figure 1.1a) or multiple embeddings (Figure 1.1b) – depending on the number of its senses [5]? While this single- vs multi-prototype representation discussion has faded in popularity following the introduction of contextualized word representations [6, 7], it is still a question worth pursuing an answer to. Although contextualized approaches are capable of modeling any number of senses, we are seldom able to decode what the determined senses are, and whether the internal representations of contextualized models correspond to senses as intuited by humans. The discrete nature of multi-prototype representations, while posing a technical challenge during training, offers benefits in the form of interpretability and insights into how neural models learn language. Better understanding of what linguistic knowledge is embedded in models can either lead to understanding where they fail, guiding practitioners to further improvements or even, albeit less likely, lead to humans discovering new insights about complex interactions in language. Most importantly, improvements upon the multi-prototype variant, such as the idea of using shared word representations (Figure 1.1c), could yield advances such as automatic synonyms induction or even cross-lingual transfer of sense inventories from languages with plentiful training data to low-resource ones.

One of the main downsides of the multi-prototype approach to word representations is that prototypes corresponding to related senses of different words do not share information throughout the training process. As word representations for more frequent tokens are exposed to more contexts, it is natural that their distributional vectors would encapsulate a better estimate of the overall distributional information for that word, thus making it easier for downstream models to determine which of the contextual patterns is applied to the downstream task. However, if we learn multiple representations for each word, it follows that each learned representation will be estimated based on fewer contexts than if we used a single-prototype approach. This, along with the requirement for disambiguating which sense occurs in a context, has been the bane of multi-prototype approaches becoming widely adopted. However, recent advances in inducing sparsity in neural networks [8, 9] opened a new avenue towards learning *shared* word representations [10] – where a sense shared by multiple words would be tuned based on its occurence in every context, regardless of the concrete surface form which will benefit the individual representation of each of the words through exposing it to more contextual patterns.

Apart from learning adequate word embeddings, with wider adoption of NLP models in high-stakes decision scenarios, a need has arisen for their opaque nature to be unveiled. Ever since its introduction, the attention mechanism [11] seemed to provide a glimpse into the inner workings of neural models – through assigning scalar weights to hidden representations, we could perhaps understand which components of input are crucial for a task, and which relevant patterns are perhaps not recognized by the network. However, this practice came under question when experiments have shown that attention should not be trusted [12], and researchers have started advocating for specialized saliency methods to be used instead [13].

Based on related work, it is inconclusive whether learning multi-prototype representations of words could offer any benefits when compared to contextualized word representations. Apart

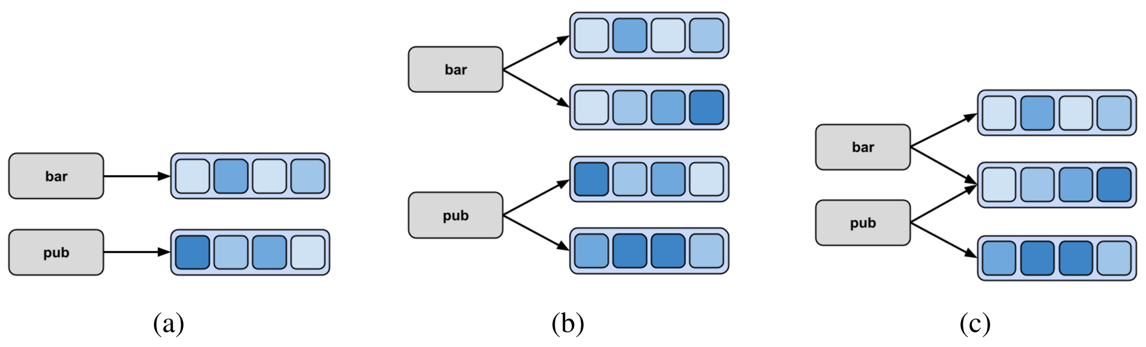


Figure 1.1: In the single-prototype approach (a), each word is assigned a single dense word representation. The multi-prototype approach (b) takes into account the existence of polysemous words and aims to assign one dense representation for each sense of the word. In the shared representation approach (c), the goal is to reuse the same sense representation for all of the words which share that sense – reducing the issue where senses infrequent for some words are exposed to fewer contexts.

from obtaining intepretability through explicit sense representations, we believe that learning a sense inventory could have implications for cross-lingual transfer larning as senses are independent of language, and could be adapted to any language by simply learning the word to sense mapping. We also find that current work on using recurrent neural networks identifies a number of key issues in convergence, mainly with respect to hidden representations not differing from one another, for which no attempt has yet been made to resolve. This thesis aims to fill the research gaps by investigating the language understanding and semantic composition capabilities of recurrent neural models. We start by improving upon the interpretability of recurrent neural network based NLP models when applied to sequence classification tasks, where we observe a considerable boost to the faithfulness of interpreting model decisions with the attention mechanism. In parallel, we perform a study that aims to determine whether learning shared word representations, a paradigm extending upon the multi-prototype representation approach, is feasible and whether it yields improvements on downstream tasks. Shared word representations require a high level of sparsity when selecting senses due to the number of sense representations being higher than the dimensionality of the sense representations, which would otherwise allow for trivial solutions through the canonical basis. Regrettably, such a level of sparsity has proven to be too high for such a model to be learned as part of an end-to-end autoencoder and as a result, the proposed models have demonstrated difficulties in improving upon competing approaches. Finally, we take the lessons learned from these two lines of research and present a model combining recurrent neural networks and a novel interative recursive attention algorithm which improves upon the semantic composition capabilities of commonly used recurrent models, providing both competitive results as well as interpretability on sentiment analysis benchmarks.

More concretely, this thesis addresses the following research questions:

- **(RQ1):** How well do recurrent neural networks capture the intricacies of sentence-level semantics and where do they fall short?
- **(RQ2):** Are multi-prototype word representations a good fit for neural models, and can such representations be beneficial for downstream tasks?
- **(RQ3):** Can we devise a novel model based on recurrent neural networks that improves both upon their representational capabilities as well as interpretability?

The aforementioned research questions constitute some of the issues that bothered me the most throughout my journey of understanding neural models for NLP and I strongly believe the answers (or even the followup questions) produced in scope of this thesis will help others taking the same path.

1.1 Contributions

The research objective of this doctoral thesis is to analyze the inner workings of recurrent neural network based algorithms for processing natural language and determine possible areas of improvement. My main goal of this thesis was to leverage the attention mechanism which is, in my opinion, one of the most transformative recent innovations in neural networks, to both bring better understanding of the capabilities of neural models and improve upon them by incorporating targeted inductive biases. The prospective original scientific contribution of this thesis consists of:

1. Empirical analysis of the convergence of recurrent neural network algorithms on text sequence modeling tasks with respect to different input word representations;
2. An algorithm for learning word representations serving as input to text processing models based on contextualized word representations;
3. An extension of the recurrent neural network model for processing text sequences with mechanisms for processing linguistic phenomena such as polysemy, semantic composition, and coreference.

1.2 Thesis structure

This thesis consists of seven chapters, including the previous introduction. In Chapter 2 we provide a brief tour through relevant developments in deep neural networks for natural language processing with respect to learning word representations both of multiprototype and contextualized type, as well as in model interpretability, covering both saliency- and attention-based methods. We then describe the body of research constituting the contributions of this doctoral thesis, starting by the main contribution of the thesis in the work on interpreting the pitfalls of neural models for NLP in Chapter 3. We continue by covering the experiments performed on learning shared word representations, a specific variant of multi-prototype word representations in Chapter 4. Combining lessons learned through analysing recurrent networks, we present an iterative recursive attention-based algorithm for processing language that is both interpretable and competitive with models that at that point were considered the state of the art. Taking a step back, we take a look at the larger picture presented through the undertaken work in Chapter 6 before concluding the thesis and discussing future avenues of work in Chapter 7.

Chapter 2

Background

Before delving deeper into the research done within this doctoral thesis, we will first provide an overview of the methods used in neural natural language processing (NNLP) and emphasize the problems which the undertaken work aimed to solve.

The first topic we will cover is learning word representations in NLP. While understanding language comes naturally to humans given time, some abstractions which we have internalized, such as semantic and syntactic similarity and inferring meaning of unseen words from context are not natural for machines. Paragraphs, sentences, and even words are sequences of symbols at different granularity. In order for these sequences to be processed by a machine learning model, they usually have to be converted to a machine readable, numeric format. Prior to defining the transformation from a symbol to a number, one first had to determine the granularity at which text was processed. Traditionally, the unit of choice was a *word*. Each text which was to be processed by a machine learning algorithm was first *tokenized* – transformed to a sequence of tokens. Tokens obtained by this process are not limited to words. Texts include punctuation and various artefacts, and tokenization algorithms are not perfect. Thus, the sequence of tokens was usually subjected to rigorous filtering that aimed to remove everything considered irrelevant for the task at hand. It was also common to discard tokens with very low frequency in a corpus,

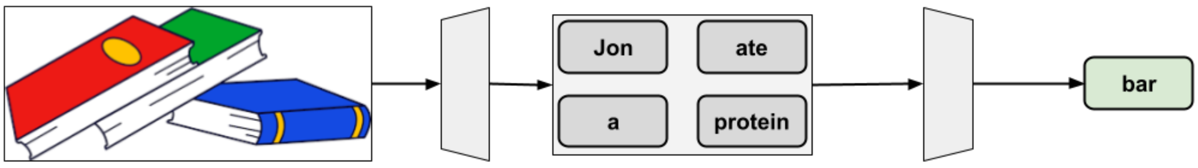


Figure 2.1: An example of a standard NLP pipeline. The textual data first has to be preprocessed to obtain a sequence of tokens. The seuqence of tokens can then be passed through a machine learning model to perform a specific task. In this example, we show the task of *language modeling*, where based on *N* preceding, context words the machine learning model has to make a prediction as to which word is the following one.

as they took up space and rarely produced value. Not only infrequent tokens were targeted though – the peak of the Zipf [14] curve was also considered irrelevant, as word types such as conjunctions and prepositions, while frequent, rarely offered any discriminative value for standard NLP tasks. These words were filtered out through curated *stop word* lists. Once adequately filtered, one had to determine how many unique tokens should be considered in the machine learning model. The considered words formed a *vocabulary*, and the size of the vocabulary is a common hyperparameter in NLP models.

After the set of considered words was determined, what remained was to determine how these symbols should be mapped to a numeric format. The leading paradigm of the pre-neural era was one of *sparse* representations. Each dataset instance was represented as a sparse one-hot vector in $\mathbb{R}^{|V|}$. Each element of the vector was zero if the corresponding word did not occur in the instance, and non-zero otherwise. Values assigned to non-zero positions differed between approaches. In simple Bernoulli representation, if a word occurred it was assigned a 1. Multinoulli representations considered word frequency, and the position was assigned the frequency of the corresponding word in the instance. However, for some algorithms, indicating relative frequency of the word in a corpus was also determined to be relevant, and approaches such as *TF-IDF* and *naive Bayes* weighting became prevalent.

The major downsides of the sparse approach were (1) the semantic similarity between words was not in any way encoded in the represenations and (2) word order was disregarded when computing the sequence representation. While the latter issue was somewhat alleviated by capturing local word order through *n*-grams, progress in accurately representing semantic similarity advanced rapidly with the introduction of neural vector space models (VSMs) of word meaning.

2.1 Vector Space Models of Word Meaning

Progress in encapsulating similarity became pronounced with the resurgence of neural networks. Vector space models were first introduced in [15] for representing documents. The authors represented each document as a vector of index term weights, with the key to improving retrieval of documents related to a search request being the quality of the index vocabulary. The goal of the work was to find an index vocabulary that maximizes the distances between entities (documents) in the vector space – demonstrating that indexing performance inversely correlates with vector space density. The performance of VSMs for document representations inspired work on VSMs of word meaning [3, 16]. Where one can represent a document though occurences of key index terms, one can also represent a word with the most frequent first-order context terms. Such approaches were first proposed [3, 17] to treat sense induction, and the representations were learned through techniques such as SVD on cooccurrence data from raw texts.

Parameters of initial VSMs of word meaning were computed through cooccurence matrix decomposition, by constructing a low-dimensional representation of distributional statistics. In contrast, deep learning allowed to learn d-dimensional word vector representations $w_i \in \mathbb{R}^d$ with gradient descent [18], enabling neural models to select and encode relevant contextual information themselves. Such, *predictive* approaches to learning word representations have demonstrated an improved capacity over their matrix decomposition based counterparts, which we believe to be caused by the aforementioned freedom in feature selection. In the following sections, we will provide a brief overview of neural approaches to VSMs.

2.1.1 Approaches

A Neural Probabilistic Language Model

A change of the established paradigm of using count-based representations of text data started with the introduction of a neural probabilistic language model [19]. The model was the first to introduce a neural model which, along with the network, also optimized the parameters of *word embeddings*. The model was applied to the task of language modeling, predicting the subsequent word $w^{(t)}$ given N preceding, context[*] words $\{w^{(t-N-1)}, \dots, w^{(t-1)}\}$. Given N d-dimensional word embeddings, the model applied a feed-forward neural network on the concatenated embeddings to obtain probabilities for the next word over the entire vocabulary. Trained with stochastic gradient descent on the cross-entropy objective, the model demonstrated

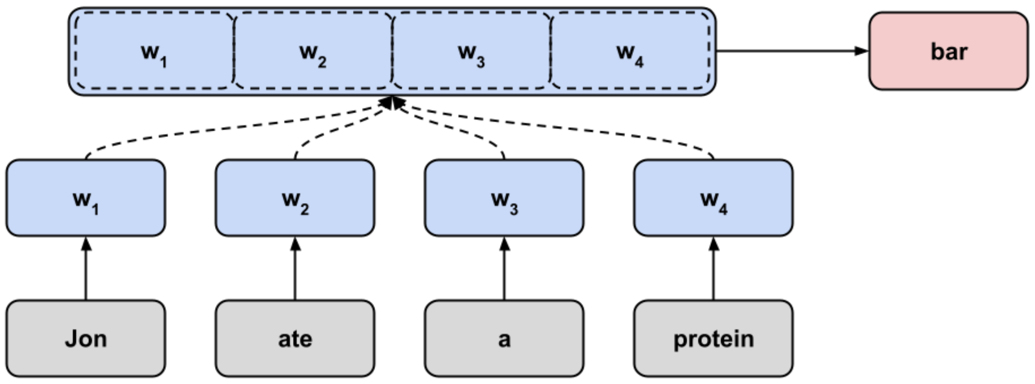


Figure 2.2: Vector space model approach to language modeling. The context size (number of preceding words) had to be kept fixed throughout training and testing. Next word prediction is done over the entire vocabulary. Each word w_i is embedded in $\mathbb{R}^d$.

significant improvements over competing n-gram language models. The territory where the neural approach excelled was modeling longer contexts – n-gram based approaches relied heavily on overlapping sequences, where word embeddings provided notions of similarity through their shared vector space. Interestingly, one of the ideas for future work in [19] states that assigning a

[*]Note that context, in general, need not refer merely to preceding words. Depending on the source, the meaning of context can refer to either preceding or following the target word.

single embedding to each word might not be the best course of action due to polysemous words, hinting towards the idea of assigning multiple representations (prototypes) to each word. Subsequent work focused on either improving downstream performance of learned word embeddings [20, 21], reduce the costly training time [22], or use neural models with inductive biases that better fit natural language [23].

Improvements Through Multi-Task Learning

A further improvement in learning word representations was shown in [24], where the authors leveraged convolutional neural networks and multi-task learning to simultaneously learn representations which can be used for a number of tasks, such as part-of-speech (POS) tagging, chunking, named entity recognition (NER) and semantic role labeling (SRL). Moving from the fixed-size window representation used in prior work, the authors use Time-Delay Neural Networks (TDNNs) [25] to allow for multi-layer processing, first extracting local features in the lower layers, while the upper layers were exposed to progressively wider context.

Cooccurrence is All You Need

In all of the previous work the word representations were learned as part of a (deep) network performing a concrete NLP task. Thus, some parameters required for the task were not used A major breaktrough at the time was the first model able to learn word embeddings at scale – *word2vec* [26] offered an efficient and effective way of producing representations which encapsulated both syntactic and semantic similarity between words. The main difference with respect to prior work came from treating word embeddings not as just another component of a model, but as the sole goal of the optimization process.

Once learned, the vector representations of each word could be stored and used to initialize the embedding matrix of a neural model, providing performance improvements across the board. Apart from quantitative improvements, word2vec vectors also highlighted interesting properties of the learned vector space – some linguistic notions such as number, gender, and semantic relations such as country-capital could be applied by moving along some fixed direction (Figure 2.3).

Other approaches to learning word embeddings soon followed [27, 28, 29], each with different theoretical properties and optimization techniques but comparable performances when controlled for hyperparameter optimization [30]. The similarity in performance of these models was further explained by establishing a link between popular techniques of learning word embeddings and matrix factorization techniques [31], where each technique introduced a specific bias to the cooccurrence matrix.

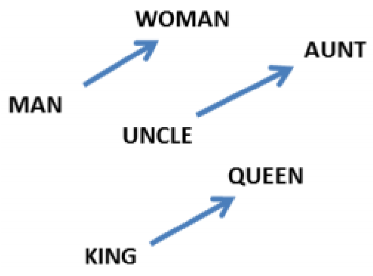


Figure 2.3: Some linguistic relationships between words could be represented as fixed transformations (e.g. translations) in the induced vector space. The example demonstrates gender as a translation in the 2-D projection of skip-gram embeddings. Further examples of this phenomenon can be found in [4].

Implications

If any doubts existed with respect to efficiency of neural word representations, the introduction of word2vec models cemented the transition to vector space models of word meaning and neural networks in NLP. Despite the performance gains which dense vector representations brought to the table, some glaring issues remained. Noted in early work [19], models which assign a single dense vector representation to each word (henceforth single *prototype* models) do not account for polysemy and homonymy. Naturally, one would think this to be an issue. It would certainly not be possible that two words could be equally well represented in a fixed size vector[†] when one of them is monosemous and the other highly polysemous? Practice would have it otherwise. It has been hypothesized and to an extent experimentally shown [10] that a single word prototype stores all senses identified in a corpus as a mixture, although this hypothesis falls apart when the number of senses grows [32].

2.2 Word Representations and Polysemy

Ever since [33] it was understood that discriminating senses of words dependent on context is a issue which at some point will have to be tackled by NLP models. In the following chapter we will present a detailed overview of conteporary models of models which address the issue of polysemy through specialized techniques of word representation.

2.2.1 Approaches

Multiple Word Prototypes

The first proposed solution [34] to the issue of polysemy leveraged a simple idea: if each word can have multiple senses, simply model all of them simultaneously. The approach belonged to

[†]The famous statement by Ray Mooney in 2014.: "*You can't cram the meaning of a whole %&!$# sentence into a single $&!# vector!*" could as easily be applied to all *senses* of a word.

the class of *prototype* models, widely studied in psychology [35, 36, 37], where each concept is represented with an abstract prototype, in contrast with *exemplar* models, where concepts are represented by sets of observed instances. Thus, instead of a single dense representation, each word was modeled as a mixture of von Mises–Fisher distributions. This approach (henceforth *multi-prototype*) allowed the model to keep a separate embedding for each sense.

The caveat here was that that the model required the capacity to determine the sense of a word based on context in order to choose the adequate prototype, a task difficult when annotated data is lacking. Annotating data for word sense disambiguation (WSD) is specifically expensive as it requires annotators determining which sense label out of a large sense inventiry is the best fit for each word in a dataset. Considering that modern NLP models are trained on dataset consistently composed of more than a million tokens, the cost of such an annotation becomes evident. The solution to the unsupervised task was an unsupervised algorithm – the prototypes for each word were determined by clustering the first-order contexts of those words in a corpus. Intuitively, the distributional hypothesis [1, 2] should apply to polysemy as well – if you know a word by its context, you should be able to deduce a sense in the same manner. Nevertheless, the multi-prototype approach demonstrated admirable improvements on word similarity testbeds. However, apart from evaluating the new approach on standard word similarity datasets, datasets which target polysemy were non-existent and authors had to rely on qualitative evaluation, which is known not to be the most reliable.

Following up, [38] extended the multi-prototype approach to include global contextual information, designed to alleviate ambiguity and act as sort of a prior in cases where local context is not sufficient to determine the sense. Apart from performance improvements, the authors also provided an important step forward in evaluating multi-prototype word representations: the SCWS dataset. In the SCWS dataset, word similarity was evaluated based on *context* and not the word tokens alone, introducing a quantitative testbed for polysemy.

Influence of Word2Vec

Both of the aforementioned multi-prototype models were initialized offline through the use of clustering algorithms and then trained with the language modeling framework introduced in [19]. Naturally, once the efficient and effective word2vec [26] framework was introduced, adapting it to the multi-prototype case was straightforward. In [39], the authors first train the skip-gram model, then exploit synset annotations present in Wikipedia to train a supervised word sense disambiguation task and, finally, jointly learn sense vectors from the now labelled corpora. While powerful, the approach has a major downside – it requires a resource with annotated word senses such as WordNet [40] or OntoNotes [41]. Any model that relies on such resources has to rely on their continuous maintenance and exhaustiveness, which is a requirement rarely fulfilled for languages other than English.

Continuing the extension of word2vec approaches, [42] introduce a multi-sense extension of the skip-gram model (MSSG), which jointly learns sense and word representations of each word. The model determines senses based on clustering instead of using an external resource for scaling purposes, and demonstrates improvements on various testbeds. Most notably, however, the model introduces a *non-parametric* variant of multi-prototype methods. Another major downside of existing approaches to polysemy was the fact that the number of senses K had to be determined apriori and set fixed for all words. Non-parametric approaches to multi-prototype models allow for the number of senses to vary and be determined by the model – if the similarity of a new context is not higher than a predefined threshold, a new sense is created.

In further work, [5] continued the thread of non-parametric models, leveraging the Chinese Restaurant Process to handle the variable number of sense representations per-word.

While well motivated, non-parametric models are notoriously difficult to train and suffer from the rich-get-richer problem, where due to the online nature of models a small number of clusters is assigned the majority of contexts.

Turning back to cluster-based models, [43] first performed LDA on context representations to obtain word topics and then used Gibbs sampling to assign topics, which act as sense labels, to each token. Once words are paired with their sense representations, it is straightforward to apply the skip-gram training procedure and learn both word and sense representations simultaneously.

Shared Sense Representations

While all prior work handled the issue of polysemy through assigning multiple prototype representations to each word, one downside was that sense representations which refer to the same concept were learned separately – even though *airplane* and *plane* refer to the same concept, the contextual information will not be shared between the distinct surface forms. In inspired work, [10] propose applying an offline sparse coding method to a pretrained word embedding matrix to identify a set of core concepts named *discourse atoms*. Discourse atoms are obtained by sparsely decomposing the embedding matrix, representing each word embedding as a sum of k discourse atom representations, which can broadly be thought of as senses. The representations of discourse atoms are obtained through a decomposition based approach:

$$\underset{\text{Embedding matrix}}{\mathbf{E}} \approx \underset{\text{Sparse coefficients}}{\mathbf{L}} \cdot \underset{\text{Discourse atoms}}{\mathbf{A}} \tag{2.1}$$

The constraint which enables non-trivial decomposition solutions is the sparsity rate of the coefficient matrix L. Since at most L coefficients are non-zero for each word, we ensure reuse of discourse atoms, allowing us to identify words that share senses. If we are able to identify words which refer to the same sense, we can then distribute contextual information between them, resulting in improved representations of each individual word. This approach has a major downside – the decomposition process has to be done online, and once determined, the word → atom mapping is fixed. Coupled with the fact that loss of information is inevitable, the method has had little practical impact. Regardless, the authors presented a very interesting finding: as each word embedding can be (linearly) decomposed to a sum of atoms, this implies that classic word embedding algorithms such as GloVe [28] and word2vec [26] are both able to recognize multiple senses of a word and conflate them in a single vector in a manner that can be retrieved through decomposition based approaches. The *linearity assertion* (2.2) states that each word can be decomposed as a linear sum of the disjoint discourse atoms (senses) it is composed of.

Nonzero sparse coefficients

$$v_w \approx \lambda_1 \; v_{s_1} + \lambda_2 \; v_{s_2} + \dots \tag{2.2}$$

Discourse atoms

The assertion was put to a test with a simple experiment: train word embeddings on a corpus. Then, replace all occurrences of two monosemous words (w_1;w_2) with a single, new pseudoword ($w_{1;2}$). Train the embedding algorithm on the new corpus, keeping all unchanged embeddings fixed. The new word is clearly polysemous, with its two major senses w_1 and w_2. In their experiments, the authors demonstrate that the pseudoword embedding has a high cosine similarity with each of the original word embeddings, despite them being learned separately.

Despite the nice theoretical implications, the linearity assertion was soon afterwards shown not to hold when the number of senses a pseudoword was composed from is large[32]. Nevertheless, further work has shown that in most cases, informations about senses are well represented in a single vector embedding, *as long as the sense is frequent* [44]. The authors also argue that even if rare senses are not well represented, that might not be an issue as rare senses are often not as relevant for downstream applications Nevertheless, we believe there is merit in representing even the rare senses of words, mainly for the fact that language is constantly evolving [45], a sense of a word that was once rare (e.g. *mouse* in the pre-computer era) could become the dominant one in a brief period of time or that senses rare in one domain could be very important for a different domain, and disregarding a sense entirely could be detrimental for transfer learning performance.

2.3 Contextualized Word Representations

Following the introduction of Transformer models [46] and subsequently BERT [7], NLP practitioners moved from traditional word embeddings to *contextualized* word representations. While the prior paradigm was to pretrain only the word representations, which were then used to initialize the word embedding matrix of a new model, it has been shown that there is a lot of merit in reusing an entire model trained on a variant of language modeling. The contextualized

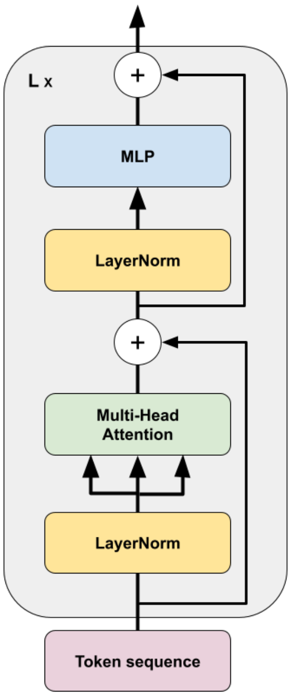


Figure 2.4: The Transformer encoder. Each network consists of L identical layers. The input token sequence consists of subword units [47]. LayerNorm denotes the layer normalization operator [48], while Multi-Head Attention [46] is dot-product attention applied multiple times in parallel over a segmented hidden representation. MLP denotes a multi-layer perceptron.

word representation paradigm contains two major conceptual deviations from previous work: (1) instead of obtaining input token[‡] representations, the goal is to obtain an entire pretrained network by leveraging a task based on language modeling and (2) instead of representing the meaning of a word in isolation, such networks model the meaning of each word in *context*, compositionally combining information from other relevant words in the sentence. The outputs of these pretrained Transformer networks are hidden states of all network layers, the uppermost ones traditionally being used as inputs of classification networks trained on downstream tasks. We refer to hidden states as *contextualized* as they have been exposed to information content of all other tokens in the input sequence – thus the information in hidden states no longer pertains to the word meaning in isolation but rather to word meaning *in context* of the input seqeuence.

[‡]When discussing Transformer architectures, tokens are in general subwords obtained through byte pair encoding techniques, in contrast to standard word tokens used in RNN networks.

Intuitively, if hidden states contain contextualized representations of the corresponding input tokens, this seemingly solves the issue of polysemy – the outputs of the transformer network will contain the context-determined sense of each token. However, although it is safe to assume that Transformer models perform sense disambiguation as part of contextualization, the sense representations are not explicitly stored anywhere and it is left to the practitioners to either decode them or make them explicit. In this section, we will provide an overview of attempts to decode sense information from large language models (LLMs) or make this information explicit.

2.3.1 Approaches to Polysemy

In [49], the authors analysed whether, and how, large language models help sense prediction. The authors leverage the WordNet [40] sense graph, which they model with a Graph Attention Network [GAT; 50]. They then train the models to jointly perform language modeling and sense prediction by predicting both the next token as well as its sense. Since the sense prediction task is supervised, the authors use SemCor [51], a WordNet-sense annotated corpus to obtain sense labels for tokens. The authors show that even when using a transformer model (albeit pretrained only on a small corpus) and an explicit graph network to encode sense relationships, next-token prediction on the granularity of senses remains a difficult task. When analysing the model errors, the authors determine that sense prediction fails when the language model fails to predict the correct next word, and hypothesize that one of the reasons for weaker performance is the long tail of the word-sense distribution caused by a large number of infrequent word-sense combinations.

In work that combines transformers and multi-prototype representations, [52] state that sense embeddings and contextualization techniques need not be mutually exclusive. Extending the standard masked language modeling [MLM; 7] framework, the authors hypothesize that each input token has a distinct set of *sememes*. At reconstruction time, instead of predicting the masked token, the model outputs $S \times V$ predictions, where V is the size of the vocabulary and S the considered number of sememes per token. The final probability of a token is then computed as the sum of probabilities for possible senses of that token $\mathbb{P}(w|c) = \sum_{s \in S_w} \mathbb{P}(s|c)$. To ensure that this approach produces a low entropy distribution over senses, the authors incorporate a *distinctness loss*, which enforces high probability in only one of the predicted sememes. To alleviate the *rich get richer* phenomenon, where the model could disregard multiple prototypes and simply exploit a single representation, bypassing the disambiguation task, the authors incorporate a *match loss*, where a separate disambiguation network is trained to decode sense probabilities based on the unmasked sequence. The sense probability distributions of the MLM network and the WSD network are then constrained to be similar, thus enforcing that the language model uses the full capacity of the multi-prototype representations.

2.4 Interpretability in Natural Language Processing

An often mentioned critique of deep neural models is their opaque nature – while they undoubtedly perform better than their shallow counterparts, it is not immediately clear which components of the input text infuence the models' decision the most. [53] have shown that, in the SNLI dataset [54], certain phenomena, such as negations and vagueness, are highly indicative of certain classes where they should not be. This fact is a consequence of the annotation process, where a simple way to create a contradiction was to either negate part of the premise or make either of the text fields too specific due to vagueness. The resulting overestimation of neural model performance poses a significant issue for practitioners – if standard benchmarks are not a trustworthy estimate of model performance, how can practitioners trust those models to perform well in high-risk scenarios?

The issue of trust has bothered researchers from the inception of the neural revolution. According to [55], the path to establishing human trust in machine learning models is through *faithfulness*, which determines whether an explanation for a decision a model made accurately represents its decision making and *plausibility*, which evaluates whether human experts would agree with this explanation for the concrete instance. If a model trained on SNLI was faithfully interpretable, practitioners could swiftly find that explanations provided by the models are implausible and determine the issue within the dataset.

Designing an interpretability method, however, is not as straightforward as it might seem. One possible goal would be to obtain a saliency map [56], which attributes a scalar score to each input component. If the score is high, the corresponding input is important, while if the score is low, the input is not important. Saliency methods come in a number of flavors: gradient-based, propagation-based, occlusion-based, and ones that use surrogate models.

An alternative to saliency methods is to use scalar scores, such as attention [11] as a proxy for saliency. When interpreting models by inspecting attention scores, we do not inspect ones assigned to network inputs but rather the hidden representations in the uppermost layer of the network. This disharmony between what attention actually pertains to and what we want it to pertain to has initially been overlooked by practitioners, as those scores passed the eye test and seemingly faithfully represented the model. Thus, every attention-based model seemed to be interpretable. This assumption was first put to test by [12], where the authors found that the attention distribution of trained models can be manipulated without detriment to the model performance. This worrying finding has sparked a still ongoing discussion [57, 58, 59, 60, 61, 62, 63] on whether attention can, and if so, how can it be a faithful method of interpretability, and why should it even be used in place of saliency methods [13]. Delving deeper into this discussion is however out of scope of this thesis, and we will merely provide an overview of the most commonly used saliency methods. We however refer the interested reader to read [64] for a succint overview of various lines of argument. In the standard sequence classification RNN + attention network (Figure 2.5), when interpreting the model decision through attention weights, we assume that the attention weights α_i assigned to hidden states h_i are a *faithful* measure of importance of the corresponding input token x_i.

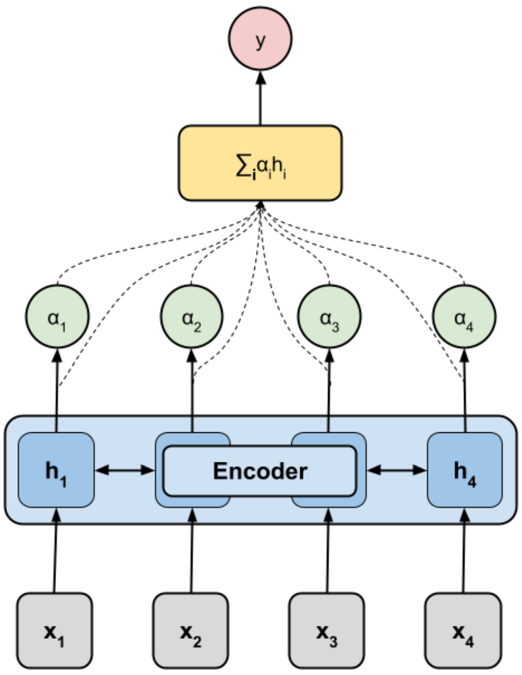


Figure 2.5: A standard single-sequence classification model. The encoder can be any contextualization network, either a RNN variant or a Transformer network, although in the scope of this thesis we consider only RNN variants. In the standard way of interpreting classifier decisions through attention weights, the attention weight α_i on the hidden state h_i is interpreted as importance of the corresponding input x_i.

On the other hand, when interpreting models through saliency techniques, specific methods are designed that assign relevance *directly* to input representations x_i. Saliency is defined as a function of the network and data $S_i = g(⁘ ,\mathbb{D})$, where $\mathbb{D} = \{(X^{(i)}, y^{(i)})\}_{i=1}^{N}$ is a dataset of size N, where $X^{(i)}$ is the input sequence and $y^{(i)}$ the corresponding class label. The function g either uses information from the network gradients (Section 2.4.1), defines its own importance propagation rules (often based on gradients Section 2.4.2) or leverages an inherently interpretable surrogate model that is trained to locally mimic the black-box deep network (Section 2.4.3). Thus, saliency scores pertain to the *inputs* directly (Figure 2.6) and their faithfulness is (*usually* [65]) not brought into question.

2.4.1 Gradient Based Interpretability Approaches

Gradient-based interpretability approaches directly leverage gradient-based information from the trained network to compute saliency scores of input tokens. While leveraging gradient

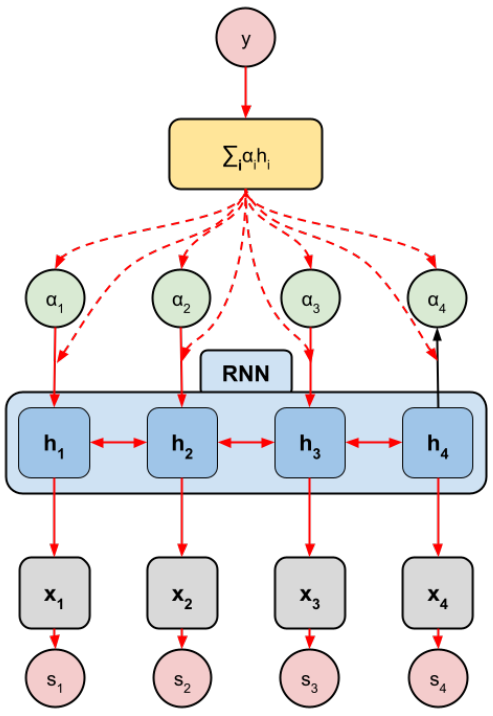


Figure 2.6: When computing saliency by either leveraging network gradients (Section 2.4.1) or using specialized importance propagation rules (Section 2.4.2), the information is propagated through the entire network. The saliency scores s_i are scalars which are directly related to the input representations.

information comes naturally, a number of nonlinear transformations in neural networks such as the logistic sigmoid and the hyperbolic tangent have near-zero derivatives on a large portion of their domains, causing issues when combined with floating point representations.

Input $\times$ grad

One of the more straightforward ways of assigning saliency scores to inputs by using the gradient of the loss function [66]. Components of the input which contribute the most *against* the decision of the model will have a high partial gradient. The authors exploit this fact, and for binary classification problems, determine the saliency of each input as the gradient when the decision of the inverse prediction ($\tilde{y} = 1 - f(x)$). Thus, the goal of the procedure is to approximate the loss of the network $\mathbb{L}$ as a linear function:

$$\mathbb{L}(\tilde{y}, f(x)) \approx w^T x + b \tag{2.3}$$

where $f(x)$ is the network evaluated at input x, and the weight vector w is obtained through a single pass of backpropagation through the network as:

$$w = \frac{\partial \mathbb{L}}{\partial x}\Big|_{(\tilde{y}, f(x))} \tag{2.4}$$

as the vector w will have a single value for each word of the input vocabulary, we can use its absolute value as the saliency measure of that word $s_i = |w_i|$. By inverting the prediction, the features with a high gradient norm for the inverted prediction will be the ones that contribute the most to the inverse prediction.

Integrated Gradients

It quickly became obvious that merely taking the gradient at a single point is not enough to determine input saliency as we have nothing to compare the relative importance to. [67] proposed incorporating a *baseline* x' to alleviate this issue, an input (or sequence of inputs) with an approximately neutral prediction value of the analyzed network. The authors determine the saliency of each input by taking an integral of the straight-line path from the baseline to the analysed input.

$$\text{IntegratedGrads}_i(x) := (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha \tag{2.5}$$

where $F(x)$ is the network output for an input x. The integral is in practice estimated with a finite number of points along the straight-line path, where the number of points is a hyperparameter of the algorithm.

The integral measures the change in prediction confidence, which if high indicates that an input is important for the models' prediction. While the authors recommend the baseline to be a vector of zero-embeddings for the case of natural language, further work has shown that there is merit in tuning the value of the baseline to ensure neutral prediction.

2.4.2 Propagation Based Interpretability Approaches

As highlighted in [67], the downside of estimating saliency as a function of the partial derivative in a single point is that local behavior of the model is not taken into consideration. Another issue with gradient-based approaches is that they are sign-sensitive [68], and as such might wrongly discard relevance of some tokens. Furthermore, when nonlinearities such as the logistic sigmoid and hyperbolic tangent are saturated, gradient values are approximately 0. Propagation-based approaches attempt to rewrite the rules of backpropagation, mainly to solve problems when backpropagating importance through nonlinear functions, where the gradients behave irregularly.

Layerwise Relevance Propagation

In [68], the authors started from the assumption that the total *relevance* in a network is constant between the network layers. Then, the task becomes to determine how relevance is propagated between layers while adhering to the *conservation law*:

$$f(x) = \cdots = \sum_{d \in l+1} R_d^{(l+1)} = \sum_{d \in l} R_d^{(l)} = \cdots = \sum_{d \in l} R_d^{(1)} \tag{2.6}$$

where R_d^l is the *relevance score* for each dimension d of a vector at layer l. The authors introduce a set of rules (dependent on which function is being propagated through) based on Taylor decomposition, thus approximating local behavior around a point of interest. Once the relevance scores are propagated to the input layers, one can determine the saliency of each input through its relevance. While empirically efficient, this approach has two major downsides: (1) practitioners have to define propagation rules for newly introduced layers and (2) in NLP, Layerwise Relevance Propagation (LRP) applies only to bag-of-words inputs, severely limiting the applicability of the model.

DeepLift

In [69], the authors determine that for ReLU networks, LRP [68] is equivalent to Input $\times$ grad [66] up to a scaling factor. Similarly to integrated gradients, the authors incorporate a *baseline* (a reference state), which they highlight as crucial for determining any insightful results for interpretability. For example, if t is the activation of some target neuron and t^0 the reference (neutral) activation of the same neuron, then $\Delta t = t - t^0$ is the difference-from-reference. The difference-from-reference is then used to assign attribution scores to each input of the analysed neuron:

$$\sum_{i=1}^{n} C_{\Delta x_i \Delta t} = \Delta t \tag{2.7}$$

where the total difference-from-reference Δt is distributed appropriately with respect to each input x_i. By using reference states and incorporating difference-from-reference in the backpropagation equations, the authors resolve the problem of backpropagating through saturated nonlinearities.

2.4.3 Surrogate Model Based Interpretability Approaches

Previous approaches have leveraged the inner workings of the trained network to determine saliency scores. A downside of these models is that they require access to the inner workings of the model, which cannot always be ensured. The goal of surrogate model based approaches to interpretability is to train a simple interpretable model which locally approximates the behavior of a complex trained model for each instance.

Local Interpretable Model-agnostic Explanations

In [70], the authors introduced Local Interpretable Model-agnostic Explanations (LIME), the goal of which is to identify an interpretable model over the interpretable input representation which is locally faithful to the original classifier. When defining the interpretable representation for text, the authors resort to the bag-of-words representation, encoding each input with a binary mask for each token in a considered vocabulary.

The training objective of LIME tries to balance between providing a good approximation $g \in \mathbb{G}$ of the original model f through the fidelity loss $\mathscr{L}(f, g, \pi_x)$ and choosing a sufficiently simple model through enforcing a complexity penalty $\Omega(g)$. Thus, the loss of the network $\mathscr{E}$ decomposes into the following two parts:

$$\mathscr{E}(x) = \operatorname*{argmin}_{g \in \mathbb{G}} \underbrace{\mathscr{L}(f, g, \pi_x)}_{\text{Fidelity loss}} + \underbrace{\Omega(g)}_{\text{Complexity penalty}} \tag{2.8}$$

One of the major downsides of LIME, however, is its significant computational overhead. A new instance of the simple model has to be trained for *each* instance of the dataset – which often proves to be prohibitively expensive as the dataset size grows. Furthermore, as the simple models use bag-of-words representations for their inputs, they once again resort to disregarding information about word order in the instances – which raises the question of whether the local approximations can even be trusted.

Taking everything into perspective, in the scope of this thesis we are mainly interested in answering the following questions: (1) can we improve upon the ability of recurrent neural network models to perform semantic composition?; and (2) can we explicitly model shared sense information in such a way that it improves the performance of models which leverage such representations? We will largely leverage interpretability methods to answer the aforementioned questions in the following chapters.

Chapter 3

Improving Faithfulness of Attention Explanations

Ever since their introduction, recurrent neural networks [RNN; 71] were prone to issues during parameter optimization – most notably learning long-term dependencies, where networks are shown to forget information past a certain window of tokens [72] and vanishing and exploding gradients, which stem from the repeated application of the recurrent state update and cause significant optimization issues [73]. Subsequent improvements upon the Elman RNN cell [74] introduced gating mechanisms to limit access to the recurrent memory and an additive expression aimed to reduce the cause of the exploding gradient issue during backpropagation. Some of these changes were shown not to be necessary [75], prompting another recurrent cell variant with a reduced number of parameters.

Nevertheless, some issues still persisted, most notably the vanishing gradient problem. These issues prevented very deep recurrent models, capping their depth between 4 and 8 layers in most practical scenarios, with each increment requiring careful tuning and a much larger dataset. These lingering issues paved the way for a switch to a novel neural architecture for processing sequences in form of the Transformer network [46]. The Transformer network offered many benefits, mainly being a good fit for training large language models (LLMs) such as BERT [7] and GPT-3 [76]. Despite the undoubtedly greater popularity and performance benefits of Transformer-based models, they have their own weaknesses in the form of requiring positional embeddings and using a subword vocabulary. In our line of work we will first attempt to unveil what is causing the known issues with recurrent networks. Concretely, we will look at one of the most basic problem in text classification – binary single sequence classification, where a passage of text needs to be assigned one of two classes. The binary classification task is one of the canonical tasks, as virtually any multi-class classification problem can be framed as a binary classification task for each class, indicating whether the input instance should be labeled with that class or not. The most frequent example of binary document classification is sentiment analysis, where the task is to determine whether a passage of text carries positive or negative polarity.

3.1 Methodology

One unexpected issue of recurrent networks became apparent from the experiments of [12, 59, 77]. The experiments have shown that LSTM networks with attention, such as the one in Figure 2.5, are surprisingly robust with respect to perturbations in the attention distribution – their prediction seldom changes, even if the attention weights are permuted, replaced with an uniform distribution, or otherwise adversarially modified. This behavior has a multitude of possible causes, however a simple explanation of why values of the attention distribution are not important would be that the hidden states themselves are very similar to one another. Indeed, it is commonly assumed that a hidden state h_t contains the contextualized counterpart of the input embedding x_t. However, this need not be the case – there is no network component that ensures retention of information from the input in the hidden state. In [62, 78], we pinpoint the causes of unwanted behavior in recurrent networks to the phenomena (Figure 3.1) of lateral information leakage (LIL; Figure 3.1a) and vertical information blockage (VIB; Figure 3.1b). We further demonstrate that by incorporating token-level regularization in RNN-based single sequence classification models the previously observed issues are alleviated, and that both faithfulness and plausibility of interpreting models through inspecting attention weights are improved. We next elaborate these findings in more detail.

3.1.1 Regularizing models

When regularizing neural models, one incorporates an additional loss term into the equation, often multiplied by a regularization *weight*. The goal of the regularization is to bias model optimization towards a certain group of solutions that either exhibit some desired behavior or are less complex. Weighing the regularization component of the loss allows us to control the relative importance of the regularization constraint being fulfilled when compared to the performance of the model on the training task.

The lack of a word-level component in the loss function of the neural network was identified as the main culprit for a weakened relationship between the hidden state h_t and input word representation x_t [62]. This weakened relationship curtails the faithfulness of interpreting attention weights α_t as an explanation of a model's decision making process. Luckily, this relationship can be strengthened by applying targeted regularization techniques which, to an extent, bind the hidden representations to the input embeddings. We will now elaborate our regularization setup when applied to single sequence classification models with the goal of improving interpretabil-

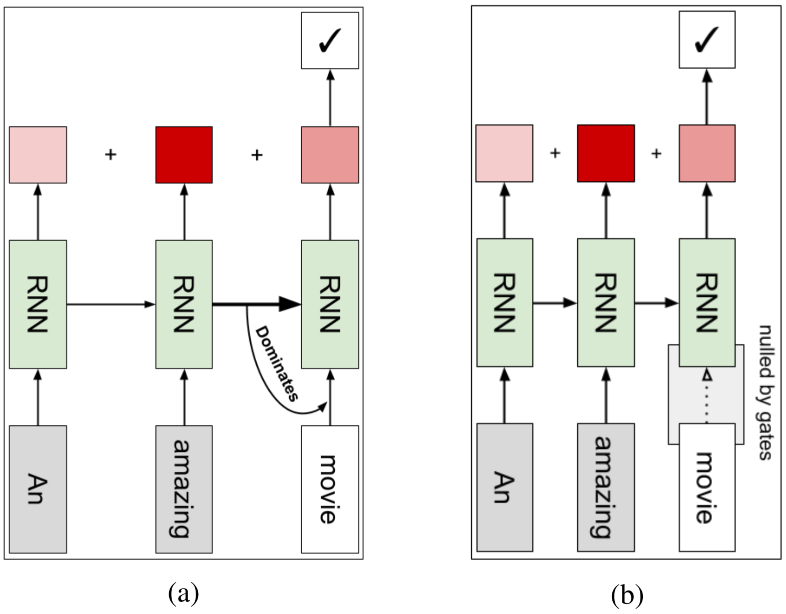


Figure 3.1: Visual intuition of lateral information leakage and vertical information blockage. In the case of lateral information leakage (Figure 3.1a), the contextual information from neighboring hidden states "overpowers" the information coming from the input representation. In the case of vertical information blockage (Figure 3.1b), the input information is completely erased by means of gating mechanisms.

ity through inspecting attention weigths and reducing LIL and VIB.

Our multilayer self-attention augmented encoder network with inputs x_t is defined as follows:

$$\mathbf{h}_t^0 = \text{emb}(\mathbf{x}_t) \qquad \alpha_t = \text{attn}(\mathbf{h}_t^{(L)})$$
$$\mathbf{h}_t^{(l)} = \text{enc}(\mathbf{h}_t^{(l-1)}) \qquad \mathbf{s} = \sum_t \alpha_t \mathbf{h}_t^{(L)} \tag{3.1}$$

where attn is either the dot-product [46] or additive [11] attention mechanism and enc is an encoder network, for example a RNN variant. The total number of layers of the encoder is L and $\mathbf{h}^{(l)}$ is the l-th layer hidden representation. We use the weights of the attention mechanism applied to the last layer to interpret a model's decisions. The sequence representation $\mathbf{s}$ is obtained as an attention-weighted (α_t) sum of the last-layer (L) hidden state representations for each element h_t in the sequence. This sequence representation $\mathbf{s}$ is then fed into a linear decoder to perform classification.

Baseline regularizations

Before introducing the regularized approaches, we will first consider simple baseline variants which explicitly encode input information in the encoded hidden representations. The two baseline methods methods which, by virtue of explicitly including input representations in the last layer hidden states, serve as a realistic upper bounds for information retention in hidden states are: CONCAT and RESIDUAL. The CONCAT method simply concatenates the word embeddings to hidden representations:

$$\mathbf{h}_t^L = [\text{enc}(\mathbf{e}_t); \mathbf{e}_t] \tag{3.2}$$

where $[\cdot;\cdot]$ is the concatenation operator. The RESIDUAL method incorporates a residual connection [79, 80] between the embeddings and hidden states. A residual connection adds the input of a neural network layer to the output, so that the output becomes $x + f(x)$, where the network layer output $f(x)$ acts as the figurative residual. Conceptually, however, the motivation for residual connections is to avoid backpropagating exclusively through nonlinear functions, which may cause the gradients to vanish or explode.

$$\mathbf{h}_t^L = \mathbf{e}_t + \text{enc}(\mathbf{e}_t) \tag{3.3}$$

We use these two methods as regularized baselines, alongside the unregularized BASE model.

L-norm regularization

Constraining the L2-norm of the difference between a word embedding and the corresponding hidden representation (henceforth TYING) promote information retention. In addition to the L2-norm, we also consider using the L1-norm and a mixture of the L1-norm and L2-norm, inspired by the ridge [81], LASSO [82], and ElasticNet [83] regularizers in regression, respectively, commonly used in regression analysis.

The training loss of a regularized classification model factorizes into two parts: the loss on the classification task $\mathbb{L}_{\text{task}}$ and the regularization term $\mathbb{L}_{\text{reg}}$. In our case, the form of $\mathbb{L}_{\text{reg}}$ depends on the type of norm used. Applying L1 or L2 regularization to single-layer networks in this case is straightforward:

$$\mathbb{L}_{\text{reg}}^{L=1} = \frac{\delta}{T} \sum_{i}^{T} ||\mathbf{h}_t^L - \mathbf{h}_t^0||_p^p \tag{3.4}$$

where we average the $||L||_p^p$ (for $p \in \{1,2\}$) of the difference between the hidden states layer $\mathbf{h}_t^L$ and the input embedding $\mathbf{h}_t^0$ across all the words. When computing the total loss $\mathbb{L} = \mathbb{L}_{\text{task}} + \delta\mathbb{L}_{reg}$, we use the hyperparameter δ to control the regularization scale when compared to the downstream task loss. A visual example of a model regularized in such a way is shown in Figure 3.2.

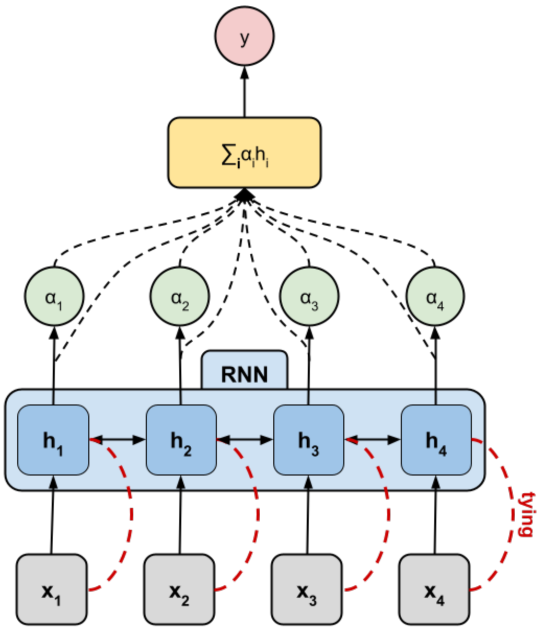


Figure 3.2: A visual example of an RNN model regularized with weight tying. The hidden states (h_i) and corresponding input embeddings (x_i) are tied according to a chosen L-norm. By tying the hidden representations and input embeddings in such a way, their similarity will increase. In turn, such regularization will improve faithfulness and plausibility of interpreting models through inspecting attention weights as they pertain to hidden states.

Masked language modelling as a regularizer

The last considered regularization method employs the masked language modelling (MLM) objective [7] as an auxiliary loss. In MLM, a certain proportion of words selected at random from the input sequence are replaced with a special mask token (`<MASK>`). The task of the model then boils down to accurately reconstructing the masked words based on the contextual information from the remaining unmasked words. It has been experimentally shown [84] that models using the MLM objective retain the correspondence between input word representations and hidden representations to a greater degree than models that optimize the standard language modeling objective of predicting the n-th word given the previous $n-1$ words.

To incorporate the MLM objective, we extend the multilayer self-attention augmented encoder network defined by (3.1) by masking the input word sequence, using the same encoder network to contextualize it, and predicting the masked words with another decoder layer:

$$\begin{aligned}\tilde{\mathbf{x}}_t &= \text{mask}(\mathbf{x}_t)\\ \tilde{\mathbf{h}}_t^0 &= \text{emb}(\tilde{\mathbf{x}}_t)\\ \tilde{\mathbf{h}}_t^l &= \text{enc}(\tilde{\mathbf{h}}_t^0)\end{aligned} \tag{3.5}$$

The hidden states $\tilde{\mathbf{h}}_t$ for the corresponding masked tokens are fed into a linear decoder, which predicts the masked word. In addition to the encoder, the embedding matrix is also shared between the MLM and classification tasks.

The decoder used for MLM does not introduce additional parameters, as we tie the weights of the decoder and the input embedding matrix, as proposed by [85]; and keep them frozen during training. In weight tying, weights of the decoder are set to the transpose of the input embedding matrix. This change effectively turns the classification task into a nearest neighbor search over the input embeddings with dot product as the metric. Both of these choices are motivated by the fact that the model might otherwise converge to a solution that does not require the retention of any information from the inputs. We will henceforth refer to models augmented with this regularization method as MLM.

The setup with the MLM objective introduces two new hyperparameters: the probability of masking a word in a sequence, denoted p_{mlm}, and the weight of the MLM loss, denoted η. Similarly to the aforementioned hyperparameter δ, the hyperparameter η plays the role of balancing the importance of optimizing the task loss and the regularization term. We keep p_{mlm} fixed at 0.15 throughout the experiments, as in [7], and adjust η with respect to the average sequence length across the evaluation datasets to ensure that the MLM loss does not dominate the optimization process. Note that because p_{mlm} is fixed, the longer the sequence, the more masked predictions the model is expected to make.

3.2 Datasets

For our experiments, we consider a diverse set of binary classification datasets used in previous work that analyzes the faithfulness of attention [12, 58, 59, 77]. The first dataset is the *Stanford Sentiment Treebank (SST)* [86], containing sentences tagged with sentiment labels ranging from 1 to 5, where 1 indicates the "very negative" class label and 5 indicates "very positive". Following previous work, we omit the "neutral" class (3) and conflate the fine-grained positive and negative classes into a single positive and negative class, respectively. We incorporate another sentiment analysis dataset in the *IMDB Large Movie Reviews Corpus (IMDB)* [87], consisting of a large number of detailed movie reviews labeled positive or negative. Delving into a different domain, we incorporate the *AG News Corpus (AGNEWS)*, a news categorization dataset. We follow the setup from [12] and limit ourselves to the two most frequent classes in "world" and "business", thus constructing a binary classification task. Another dataset belonging to the news domain is the *20 Newsgroups (20NG)* dataset, which consists of newsgroup correspondences labelled with 20 categories. We again limit ourselves to two frequent, but contrasting classes in "baseball" and "hockey". Finally, we incorporate *MIMIC ICD9 – Anemia (Anemia)* [88], a dataset from the medical domain containing patient discharge summaries from a database of electronic health records. In this case, we analyse a classification task on the *Anemia* data subset, classifying whether a summary corresponds to a patient with *acute* or *chronic* anemia.

Dataset	Vocab. size	Avg. length	Train size	Valid. size	Test size
SST	17305	17	6898	870	1818
IMDB	134463	237	17212	4304	4363
20NG	15588	180	1233	307	382
AGNEWS	47386	31	50999	8999	3799
ANEMIA	62718	1940	4653	821	1368

Table 3.1: Statistics of the five datasets used in our experiments. Vocabulary size is denoted in the number of unique words, not all of which are considered during training. The average length reflects the average number of words per instance. The last three columns contain the number of instances in the train, validation, and test splits of the dataset, respectively.

Descriptive statistics of the used datasets are summarized in Table 3.1. We limit the vocabulary size to 20000 for datasets where the number of unique tokens exceeds this amount. Tokens which are not part of the vocabulary are replaced with the unknown token (`<UNK>`), while purely numerical token are replaced with a special token representing a digit (`<DIGIT>`). For tokenization of datasets that are not already preprocessed, we used the SpaCy[*] English tokenizer. Sample preprocessed instances from each of the considered datasets are provided in Table 3.2,

Dataset	Label	Text
SST	Positive	*"just the labour involved in creating the layered richness of the imagery in this chiaroscuro of madness and light is astonishing."*
IMDB	Positive	*"i enjoyed this film very much. many korean people will feel familiar with this film because many of them have tutors... "*
AGNEWS	Business	*"business, new york (reuters) - soaring crude prices plus worries about the economy and the outlook for earnings are expected to hang over... "*
20NG	Hockey	*"this would be dumb move of the nineties lindros is big and strong but why give him a ball and chain on one leg and an anchor on the other... "*
ANEMIA	Chronic	*"admission date: <DIGIT> discharge date: <DIGIT> date of birth: <DIGIT> sex: f service: medicine allergies: penicillins/percocet/... "*

Table 3.2: Sample instances from each considered dataset. The instance data contains preprocessed text, except for trimming the punctuation whitespace on the left side.

3.3 Evaluation

Through regularizing the RNN + attention models, our goal is to reduce the effects of LIL and VIB. In order to observe a reduction in either of these, we need to measure the extent in which LIL and VIB are present in our models. To measure LIL, we use the change in F1 score

[*]`https://spacy.io`

of a trained model when the attention distribution is permuted or substituted with an uniform distribution [12]. We aim for the drop in F1 score (Eq. 3.6) to be as high as possible, indicating that the learned model is fragile with respect to uniform (*un*) and permutation (*pm*) perturbations of the trained attention distribution.

$$\Delta \mathrm{F1} = \mathrm{F1}_{train} - \mathrm{F1}_{un/pm} \tag{3.6}$$

Since the change in F1 score only measures the cases where the decision of the model changes, we proposed [78] measuring the change in Brier score [89]. The Brier score is defined as

$$\mathrm{BS} = \frac{1}{N}\sum_{t=1}^{N}(\hat{y}_t - y_t)^2 \tag{3.7}$$

where $\hat{y}_t$ is the predicted probability of the model for the “positive” class and y_t is the actual outcome for instance t in a dataset of size N. What we then aim for is to maximize the increase in Brier score, indicating that the predicted probabilities of the correct class for the perturbed model have decreased as a consequence of the perturbation.

$$\Delta \mathrm{BS} = \mathrm{BS}_{train} - \mathrm{BS}_{un/pm} \tag{3.8}$$

When measuring VIB, the goal is to measure how much (if any) information in the hidden states is retained from the corresponding input representations. The natural way to do this would be to measure the mutual information (MI) between the hidden states and input representations. However, measuring mutual information is notoriously intractable to compute in continuous form, except for rare cases where the data distributions are known [90]. Instead of exactly measuring MI, we instead discretized our continuous vector representations and computed MI in the tractable discrete case, inspired by [84, 91]. The discretization is performed in two steps: (1) we first cluster the input embeddings and the hidden representations (each of them separately) for a subset of input tokens into a large number of clusters. We then used the cluster labels in place of the continuous vectors. More precisely, we sampled 1000 words from the vocabulary and collected (up to) 1M representations of the corresponding word instances from sentences, at both the input and hidden level. For smaller datasets, where there are no 1000 words that together occur 1M times, we collected as many representations as there are instances in the dataset. We then clustered the obtained representations into $k = 1000$ clusters with minibatch k-means [92]. We obtained the vocabulary sample in two ways: as the top 1k most frequent words (**MF**), as in [84], and as a random sample (**RS**) from the scaled unigram distribution. Our choices of vocabulary sampling are motivated by the fact that the most frequent words will be seen by the model the most, and we can expect them to have well-tuned representations. Since not all words relevant for model predictions will be the most frequent words, we also evaluated

MI on a more realistic, random sample. The sample is drawn from the unigram distribution raised to the power of $\frac{3}{4}$. This transformation was found [4] to be effective in correcting the bias toward high-frequency words arising from the Zipfian distribution.

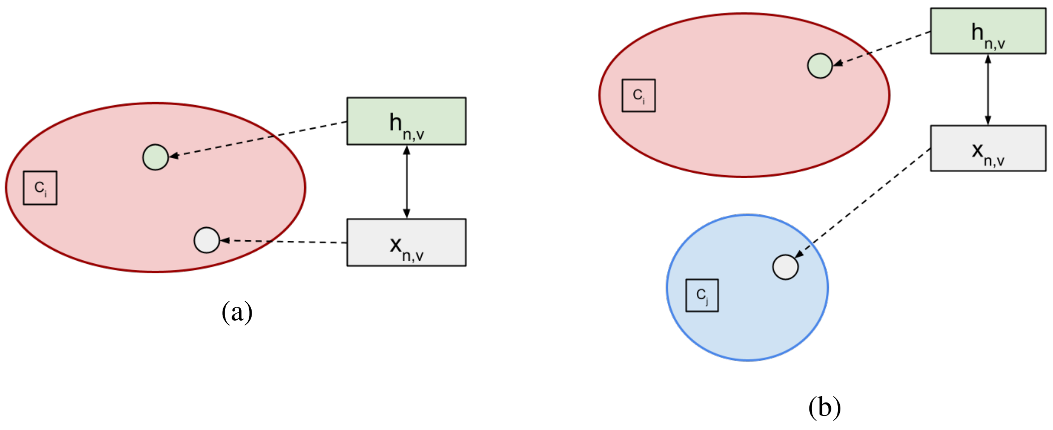


Figure 3.3: A visual intuition of estimating mutual information with discretization through k-means clustering. If the hidden $h_{n,v}$ and input $x_{n,v}$ representations for the n-th instance and word v are assigned to the same cluster (a), we consider them sufficiently similar, and such pairs increase the estimated MI. If they are assigned different cluster labels (b), we consider them insufficiently similar and they decrease the estimated MI.

3.4 Experimental Details

As throughout our experiments we have considered a wide array of different model hyper-paremeters, we will now indicate the ranges for the ones searched over as well as the ones we fixed based on manual tuning. When selecting the RNN hidden state size, we differ from [12, 58] and use $h = 150$ instead of $h = 128$ due to the tying regularization requiring the input embeddings and hidden states being of same dimensionality. Remaining hyperparameter values and intervals can be seen in Table 3.3.

3.5 Results

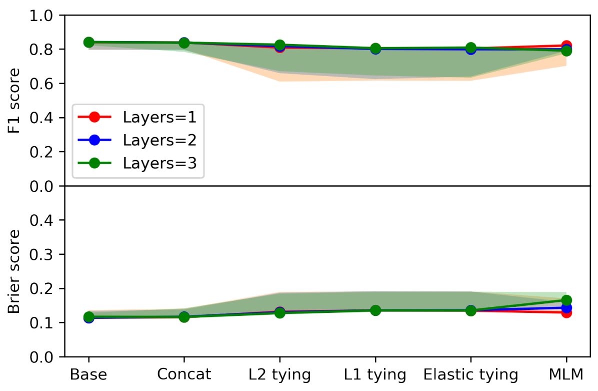
In this section, we will show how regularizing models reduces lateral information leakage and vertical information blockage. Overall, we find that incorporating regularization techniques helps faithfulness as they on average reduce the F1 score of perturbed models by 0.14, while the Brier score increases by 0.07 on average across all datasets and model types.

General hyperparameters	
Embedding dim	300
RNN hidden dim	150
Learning rate	1e−3
Grad. clipping	5
Num Epochs	5
Freeze embeddings	True
Batch size	32
Weight decay	1e−5
RNN layers	$\{1,2,3\}$
RNN cell	$\{\text{RNN},\text{GRU},\text{LSTM}\}$
Regularization hyperparameters	
Masking prob.	0.15
Masking weight η	$\{0.1,0.3,1,3,5\}$
Tying norm	$\{L_1,L_2,\text{Elastic}\}$
Tying weight δ	$\{0,0.1,0.3,1,3,10,20,30\}$

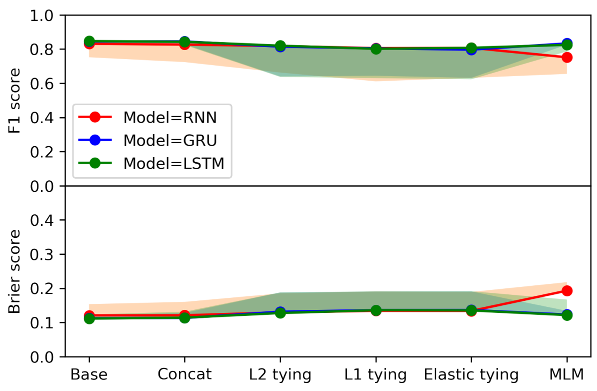
Table 3.3: Considered hyperparameters of models throughout experiments. The large search space for the tying and masking weight is due to the influence of average instance length on the regularization strength, where we observe that for datasets with longer instances on average, a smaller regularization weight is required.

3.5.1 Measuring Lateral Information Leakage

In Figures 3.4 to 3.8 we present the results of measuring lateral information leakage in the analyzed models. From the perspective of reducing lateral information leakage, we want the drop in the model's performance (which corresponds to the shaded area below the line in case of F1 score or above the line in case of the Brier score) to be as large as possible while not diminishing the classification performance of the model severely. We observe that lateral information leakage differs across models and regularization types. Namely, if the area below the graph is small, as is the case for LSTM and GRU BASE and CONCAT models (Figures 3.4 to 3.8), this indicates that perturbations of the attention weights do not affect the classifier decision, and consequently, that the explanation obtained by inspecting attention weights is not faithful. This finding is in line with the work of [73], highlighting that recurrent networks have worse convergence properties than their descendants. The weaker convergence of vanilla RNN cells causes them to overfit at a slower rate; therefore, we observe reduced lateral information leakage when



(a)



(b)

Figure 3.4: Performance across different model depths (a) and model types (b) on the SST dataset. Each line represents the F1 (top) or Brier (bottom) score of the trained recurrent model, while the shaded area represents the drop in the corresponding score score when the trained attention distribution is substituted with the uniform distribution.

compared to LSTM and GRU cells. We also notice that MLM regularization has a somewhat weaker effect on promoting faithfulness than other regularization techniques.

The SST and AGNEWS datasets (Figures 3.4 and 3.6) exhibit the smallest F1 and Brier score changes, regardless of the value of the regularization term. However, we believe this is not a symptom of the regularization methods performing worse on those datasets. It is important to note that both of those datasets have the shortest average length of instances – thus, any perturbation of attention weights will have a less adverse effect on the contribution of key tokens to the output representation. In the case of replacing the attention distribution with the uniform distribution, the intuition is as follows: for a sequence of length T, the uniform attention weight on each token will be $\frac{1}{T}$. Thus, as the (average) sequence length increases, each individual token will be less important in the output representation. Therefore, unless the number of tokens indicative of the classification grows proportionally with the length of the sequence, the output representation will have less information required for classification. For datasets with a short average length, the uniform weights on each individual token will be higher, thus preserving important information required for classification – making models trained on those datasets exhibit reduced fragility to perturbation of attention weights.

3.5.2 Measuring Vertical Information Blockage

We next investigate to what extent the information flows vertically (and to what extent is it blocked) in the considered models. We report the mutual information scores on the considered datasets in Figures 3.9 to 3.13. For the sake of space, in graphs where the MI scores are reported across different model depths, we average the scores across different models. Similarly, in graphs where the MI scores are reported across different models, we average the scores across

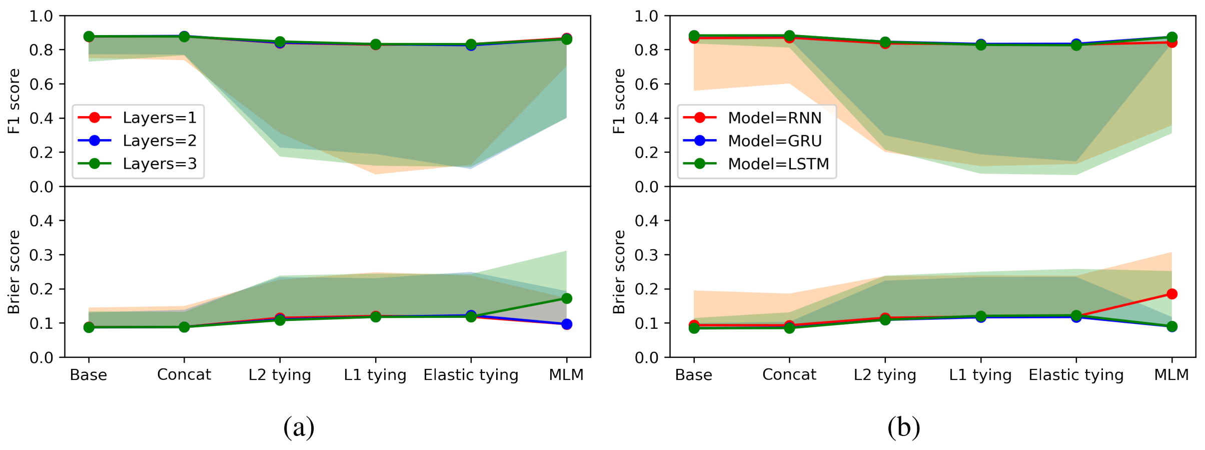


Figure 3.5: Performance across different model depths (a) and types (b) on the IMDB dataset.

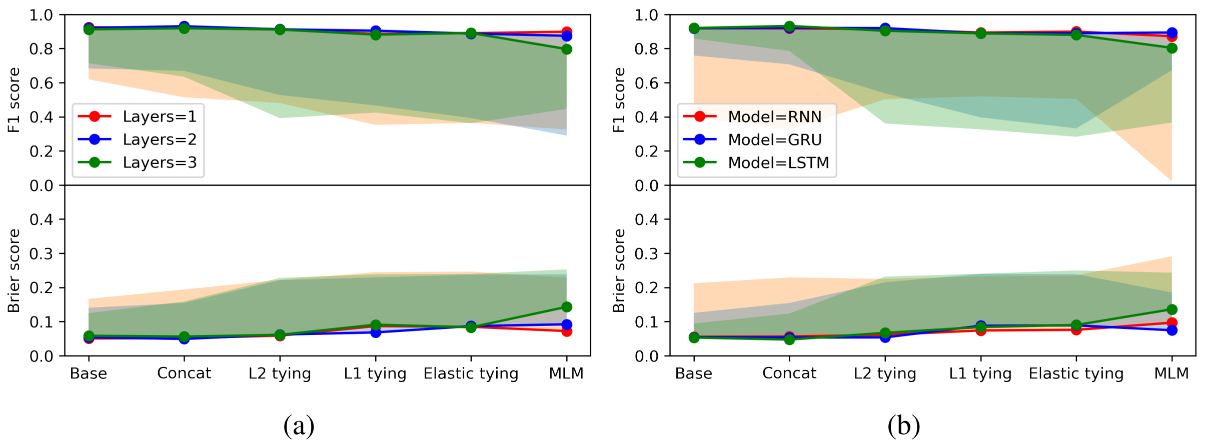


Figure 3.6: Performance across different model depths (a) and types (b) on the 20NG dataset.

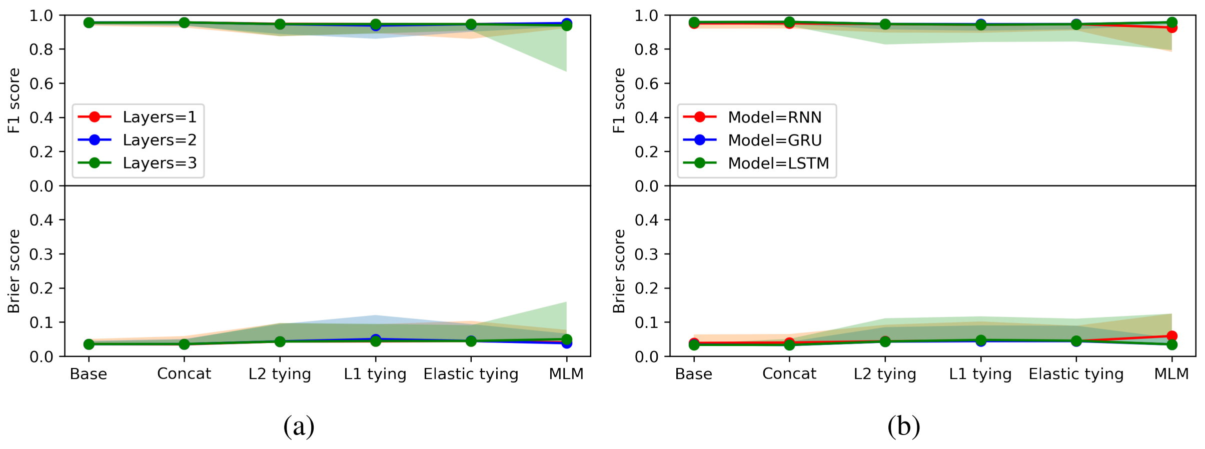


Figure 3.7: Performance across different model depths (a) and types (b) on the AGNEWS dataset.

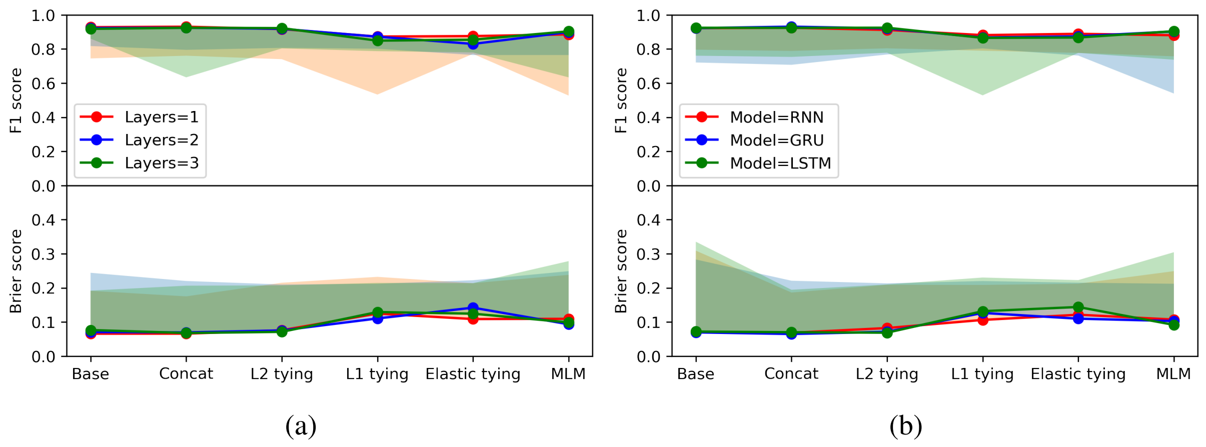


Figure 3.8: Performance across different model depths (a) and types (b) on the ANEMIA dataset.

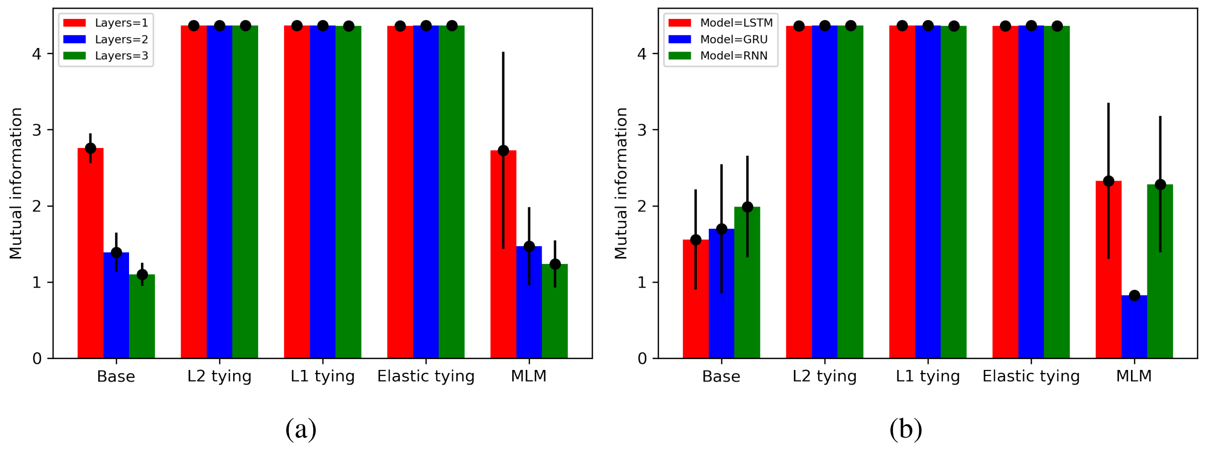


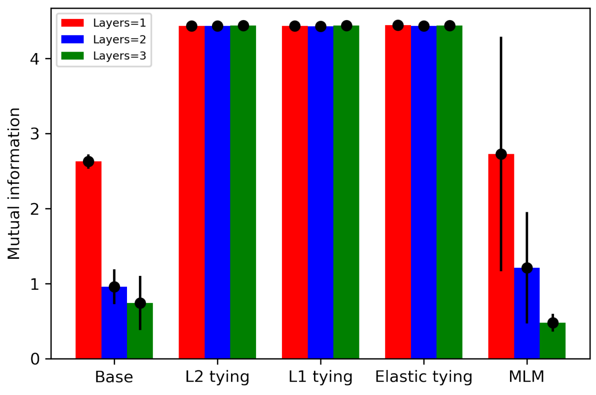
Figure 3.9: Mutual information across different model depths (a) and types (b) on the SST dataset.

different model depths for those models.

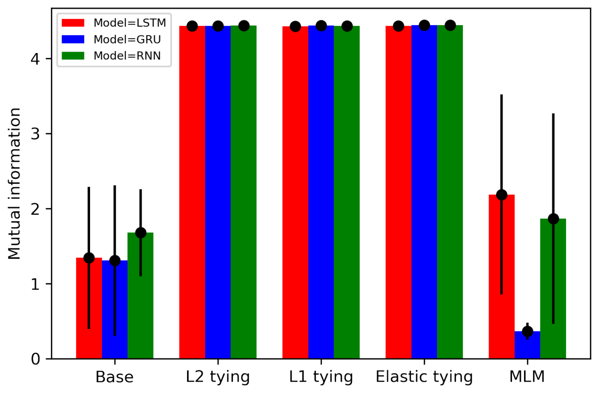
The height of each bar represents the mutual information, and the lines represent the variance. We can see that TYING regularization improves mutual information consistently and with very low variance between runs. MLM, however, does not offer consistent improvements and also exhibits high variance.

3.5.3 Visualizing the Effect of Regularization

Since the actual tangible effect of the regularization techniques is difficult to grasp from the plots themselves, we will now present visual intuiton to its effect on the models through a simple experiment. The standard intuition behind the attention mechanism is that the more attention mass is assigned to a hidden state, the more important the corresponding token is. Thus, it follows that hidden states corresponding to important tokens should benefit more to the classification decision. We will now consider a thought experiment: what if a single token was

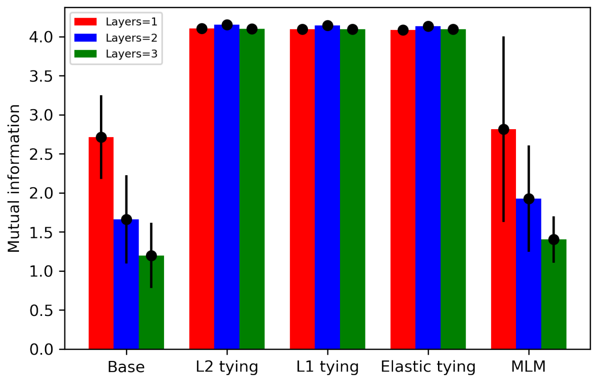


(a)

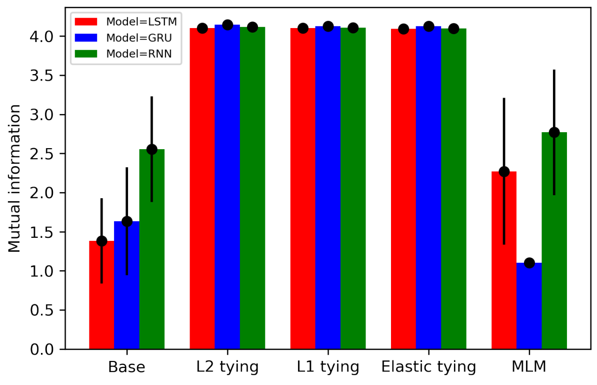


(b)

Figure 3.10: Mutual information across different model depths (a) and types (b) on the IMDB dataset.

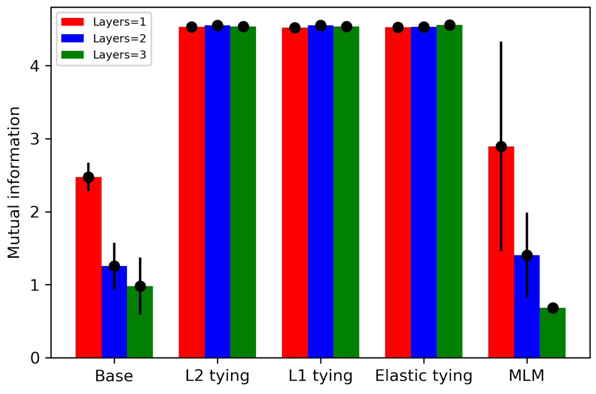


(a)

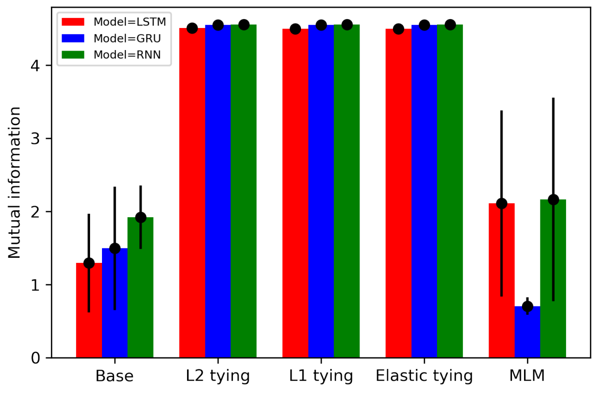


(b)

Figure 3.11: Mutual information across different model depths (a) and types (b) on the 20NG dataset.

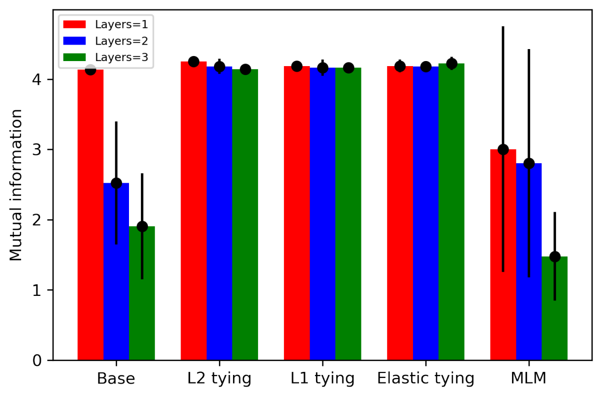


(a)

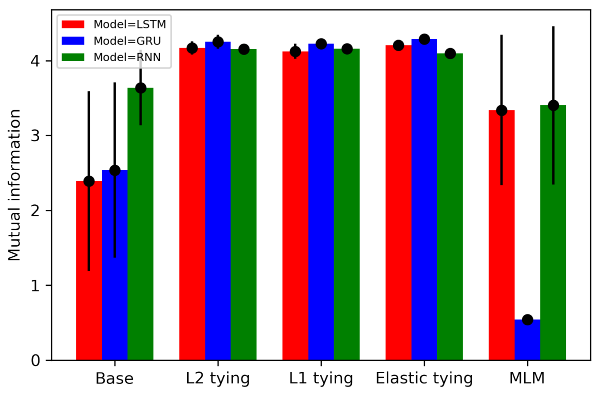


(b)

Figure 3.12: Mutual information across different model depths (a) and types (b) on the AG-NEWS dataset.



(a)



(b)

Figure 3.13: Mutual information across different model depths (a and types (b) on the ANEMIA dataset.

artificially assigned all of the attention mass and then passed through the classifier network? The assumption, following the aforementioned intuition behind the attention mechanism, would be that positive, negative or neutral polarity tokens would yield predictions reflecting their polarity. However, in unregularized models, this is not the case. In Figure 3.14 we plot the prediction probability of the positive class for binary sentiment classification on the SST dataset. For each word, we artificially assign all of the attention mass to that word and then plot the prediction probability for that artificially generated representation. We observe that, in the unregularized model, whichever token we assign all of the attention mass to, we still obtain a high probability classification. In contrast, when we regularize our models, the prediction probability for each token represents its polarity more accurately, indicating that the effects of lateral information leakage and vertical information blockage are reduced.

To expand on Figure 3.14, we now plot per-token prediction probabilities for multiple models. We sometimes omit the model classification probabilities not to clutter the plots too much. We select diverse examples (Figures 3.15 to 3.19) from the first three batches of the SST validation split.

3.5.4 Evaluating the Plausibility of Attention Distributions

When discussing interpretability of neural models [55], we care for two goals: (1) *faithfulness*, where the explanation has to be true to the model and (2) *plausibility*, where the explanation has to be plausible to human experts. We have thus far demonstrated that regularization techniques improve faithfulness, however we are still interested in their plausibility. To analyze whether explaining models through inspecting their attention weights can be a plausible method of interpretability, we analyze how the attention mass is distributed to words in models trained on sentiment analysis tasks (SST and IMDb). We leverage the opinion lexicon [93], a curated list

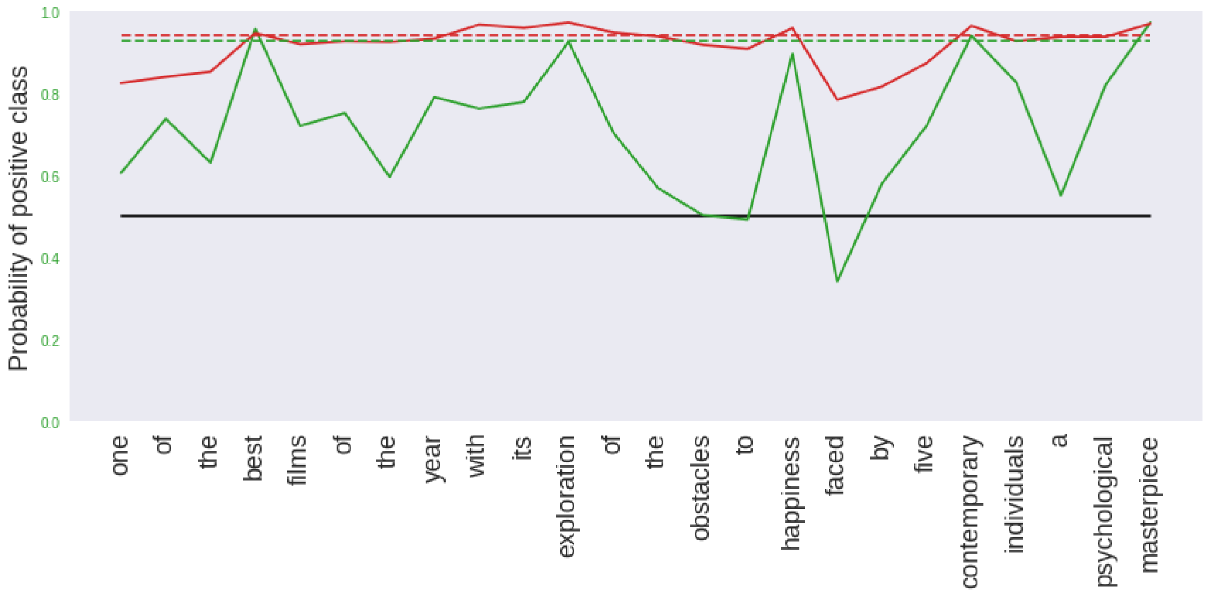


Figure 3.14: Per-token prediction probability for an example from the SST dataset for the base model (red) and a regularized (tying) model (green). The dotted lines indicate the classification probability of the model.

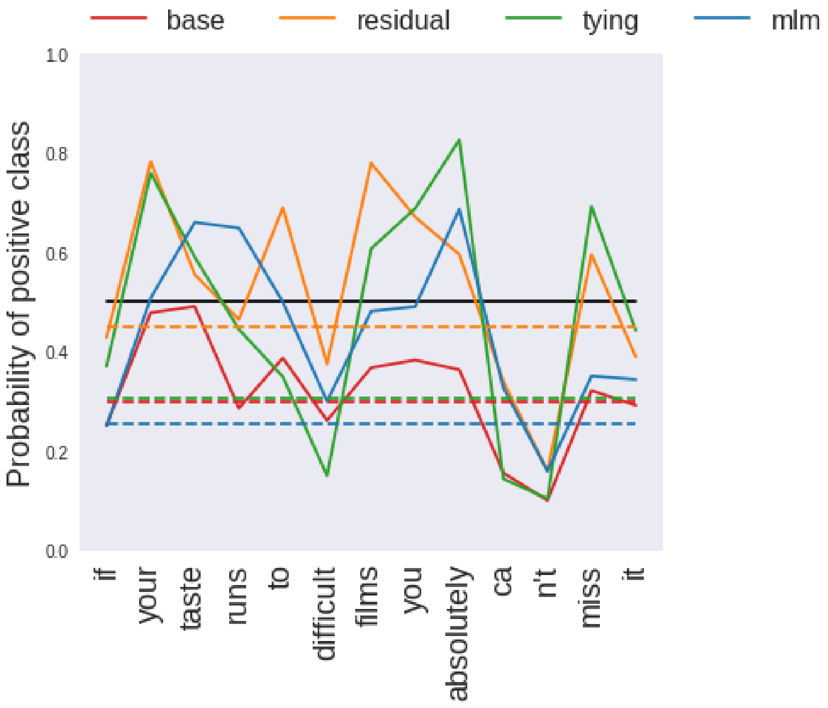


Figure 3.15: A negative example: perhaps the analysed single-layer LSTM is unable to understand even the simple nuances of language. Here the instance is classified as negative across all models only due to presence of the word “difficult”. Note that these models obtain a near 0.9 F1-score on this dataset.

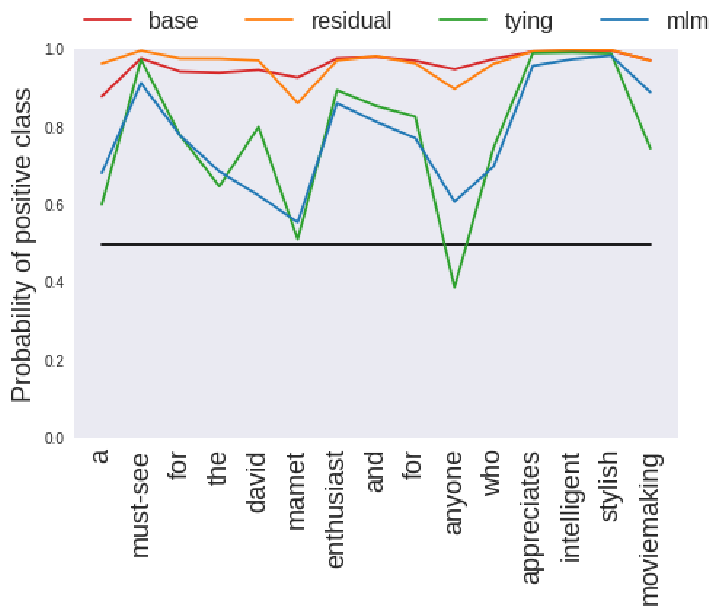


Figure 3.16: A clear-cut instance

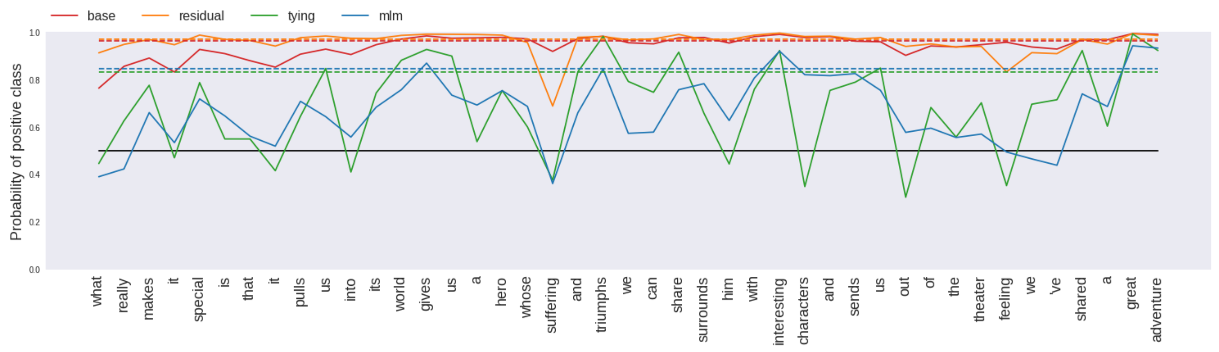


Figure 3.17: A long example which further demonstrates lateral information leakage

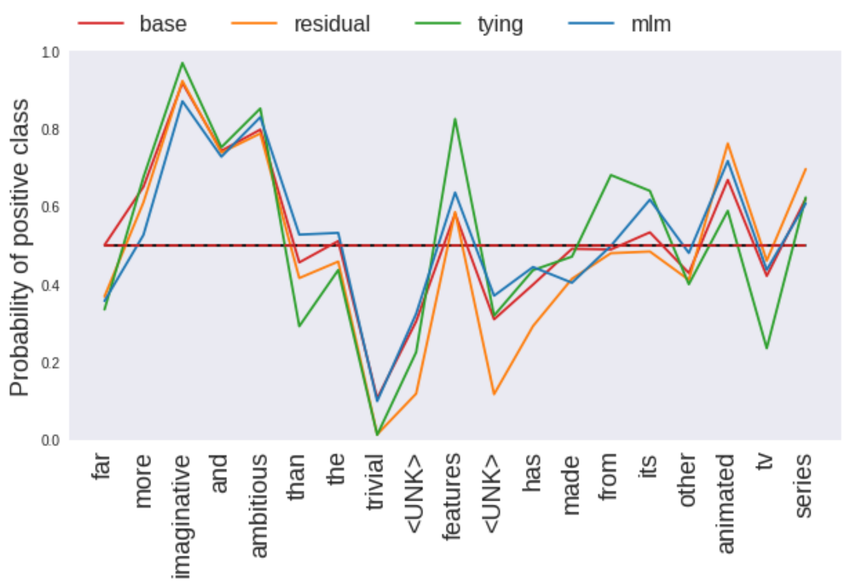


Figure 3.18: We observe that for instances where the model is not clear about the classification, the per-word probabilities are pretty similar between regularizations. We believe that lateral information leakage happens only when the model is confident in its prediction. Base model prediction confidence is indicated in this example (it overlaps with the 0.5 line).

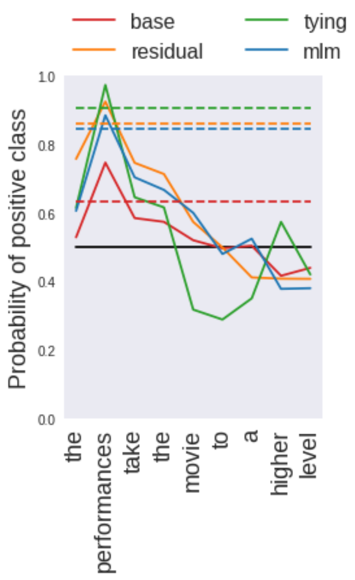


Figure 3.19: A rare example where the regularised models are more confident in the correct prediction than the base model

of 6800 words bearing positive or negative sentiment. Our aim is to investigate whether regularization methods cause the trained attention distribution to attribute more attention mass towards positive or negative words when compared to others.

Dataset	SST			IMDB		
	Pos. α	Neg. α	Neut. α	Pos. α	Neg. α	Neut. α
Base	3.970	3.980	23.944	4.391	4.032	23.517
Concat	4.699	6.109	21.086	6.570	5.576	19.698
L1 Tying	**6.036**	<u>6.231</u>	19.627	6.688	7.502	17.675
L2 Tying	5.794	6.217	19.883	<u>7.094</u>	**7.629**	17.096
Elastic Tying	5.825	**6.391**	19.679	6.733	<u>7.595</u>	17.533
MLM	<u>5.920</u>	4.383	21.591	**7.281**	7.386	17.237

Table 3.4: Average attention mass (α) attributed to positive (Pos.), negative (Neg.), and neutral (Neut.) words on the SST and IMDB datasets. Highest scores for Pos. and Neg. words are **boldfaced**, while the second highest are <u>underlined</u>. Scores reported are averages over 5 runs with different random seeds, across model types (RNN, GRU, LSTM) and depths (1-, 2-, and 3-layer models).

In Table 3.4 we show the average attention mass attributed to words annotated as either positive or negative, or not present in either inventory for each model. The values reported are averages over 5 runs with different random seeds. We observe that inducing regularization significantly increases the amount of attention mass assigned to words which are part of the opinion lexicon. We (erronously) assume that words not annotated as positive or negative are neutral, however as the opinion lexicon is not exhaustive, this is unlikely the case. However, we can assume that the *majority* of words not part of the opinion lexicon are indeed neutral, and thus the evaluation setup is still correct on average. The aforementioned inexhaustiveness is also the cause of a significant amount of attention mass being assigned to presumably neutral words, which we believe contain words specific to the movie domain and not part of general purpose sentiment lexicons.

Taking all of the presented results into consideration, we can recognize that recurrent networks have thus far possessed significant problems, in form of lateral information leakage and vertical information blockate, which influenced the correspondence between input and hidden representations. By alleviating these problems we have directly influenced semantic composition in such networks – where they previously composed a meaning representation of the entire sequence, as evident from the fact that the prediction did not change dependent on perturbations to the attention distribution, it now more accurately represents the meaning of a word *in context*.

Chapter 4

Learning Shared Word Representations

As early as [33], it was clear that a need exists for disambiguating between different senses of polysemous words. Approaches that extended the established paradigm of single-prototype word embeddings [4, 28] were plentiful [10, 30, 34, 38, 39, 42], but they never yielded widespread usage. One of the pitfalls of using multiprototype word representations is definitely the need for the discrete step of determining which of the multiple word senses is used in a certain context. Recently, pretrained models that produce contextualized word representations [6, 7, 76] have brought into question the requirement for learning sense-specific representations of words, as contextualized representations can represent a virtually unlimited number of senses.

The approach of [10] offers a possible benefit that other multiprototype did not posses. Previously, in the multiprototype word representation paradigm, each word was assigned a number of sense representations. These representations were learned independently of each other, which meant that even in the case that two words shared a sense, they would not learn the sense representation jointly. The sparse coding approach [10] seemingly solved this issue. Starting with the hypothesis that every learned word representation encapsulates sense representations for each sense of that word – and that these senses can be *teased* apart – the authors demonstrated that the sense representations can be retrieved from the learned word representation by virtue of a sparse coding algorithm [94]. In their model, the sense representations were *shared* between multiple surface forms of words. If such an idea, where recognizing a word sense would affect the representation of each word that shares that sense, could be implemented into end-to-end learning of word representations, the benefits of observing multiple contexts could outweigh the drawback of disambiguating the word sense. Apart from sharing contextual information between words which share a sense, the main benefit of eplicitly modeling sense representations, as opposed to implicitly as is the case in contextualized representations of large language models, is that since the sense inventory is independent of language, the sense embedding matrix can potentially be learned in a high resource language and transferred to a low resource one. However, this approach had one downside – it could not be trained as part of a neural network due to the nature of a sparse coding algorithm.

Following recent advances of incorporating sparsity in neural networks by virtue of methods such as REINFORCE [95], Gumbel-Softmax [8] (also known as the Concrete distribution [96]), Hard-Kumaraswamy [9], or differentiable L_0 regularization [97], including shared word representations into a neural model seemed like a possibility.

4.1 Methodology

Let us remind ourselves of the shared word representation paradigm. We have an embedding matrix $E \in \mathbb{R}^{V \times d}$, where each token from a considered vocabulary of size V is mapped to a dense embedding. These embeddings are not contextualized, and at that point we do not know which out of the possible senses of that word should be used. We aim to learn a sense inventory matrix $A \in \mathbb{R}^{A \times d_a}$ which contains all the *atomic* senses of words used in language and where the number of sense representations $A \ll V$. Our end goals here are to (1) learn a useful and exhaustive sense inventory and (2) train a model which, based on the context of a token, assigns a sense to that token.

A natural way of framing the task of learning would be to leverage a self-supervised learning approach commonly seen in practice, such as language modeling [6] or masked language modeling [7]. However, if our entire network is fully connected, we run into a problem when mapping contextualized word representations h_i to components from the sense inventory matrix a_i. If we do not limit the number of senses each word can select, our model could simply diverge to a trivial solution where, if $d \leq A$, the sense representations could simply be the canonical basis of $\mathbb{R}^d$, thus enabling the model to reconstruct any possible input representation h_i by selecting d components of the basis. Our solution to this issue is to leverage sparsity in neural models to limit the number sense representations which can be selected – zeroing out all but k connections.

4.1.1 Inducing Sparsity in Neural Networks

We will now zero in on the concrete problem we are trying to solve. After our input representations x_i have been contextualized by means of an encoder network (which can be either a Transformer or an RNN variant), we obtain a sequence of hidden representations h_i. Each of these hidden representations, by virtue of observing the entire sequence, should contain sufficient information to determine which sense of a word is used in the current context. Thus, when performing the selection operator, we should not require more than a single sense representation a_k. However, if we desire to provide some leeway for the network (which often proves to be necessary), we can relax this strict requirement and allow the network to select a subset of

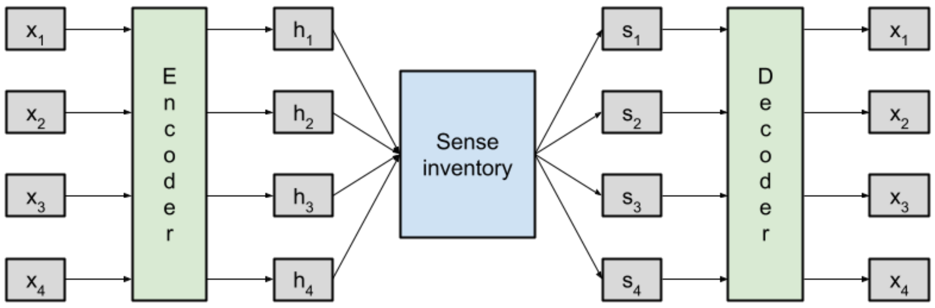


Figure 4.1: A high-level sketch of the autoencoder approach to learning shared sense representations. In contrast to denoising approaches, where a lossy transformation is applied to the inputs which then need to be reconstructed by the network, in our approach we leverage the fact that when the input sequence is sparsely compressed, some loss of information has to occur. Due to this bottleneck, both the input and output contain the input sequence and the task of the network is to learn to generalize between training instances to accurately reconstruct encoded sequences.

senses: $h_i \rightarrow \{a_k\}^K$, where K is a hyperparameter denoting the number of nonzero connections from a hidden representation to the sense inventory.

So, our goal is to learn a sparse selection operator where each hidden representation $h_i \in \mathbb{R}^h$ selects a subset of senses from the sense inventory $A \in \mathbb{R}^{A \times d_a}$. Here we abuse notation and use A to name both the sense inventory and the number of sense representations it contains. This selection operator has to be sparse as usually $A > d$ and a trivial solution would have the first d rows of the sense inventory contain the $d \times d$ unit matrix, with the remainder rendered irrelevant. Thus, we must restrict the number of senses selected from the inventory to $K \ll d$.

First we need to define the map from h_i to A. The natural approach is to leverage the attention mechanism [11], where the query is a function of the hidden representation, and the key and value representations are computed based on individual sense representations. The attention mechanism is a great fit as (1) it naturally produces a scalar value for each query–key pair and (2) the natural usage of a *softmax* nonlinearity invites a drop-in replacement in form of a discrete variant such as the Gumbel-Softmax [8, 96], or an elementwise discrete variant in the HardKuma distribution [9]. Alternatively, we also experiment with the Vector-Quantized Variational AutoEncoder (VQ-VAE) model [98], a variational approach to incorporating discrete latent variables. We will now provide a succinct overview of the main approaches to inducing sparsity in neural networks we focused on in scope of our work.

Gumbel-Softmax

The Gumbel-Softmax, or concrete distribution, is a continuous distribution over the simplex that can enable us to approximate samples from a categorical distribution. If we consider a $(k-1)$-dimensional simplex Δ^{k-1}, categorical samples from a continuous distribution with class probabilities $\pi_{\{1,...,k\}}$ would produce one-hot vectors that correspond to vertices of the simplex.

The Gumbel-Max trick [99, 100] uses inverse transform sampling in order to allow us to draw samples from a categorical distribution with probabilities π:

$$z = \text{one\_hot}\left(\arg\max_i [g_i + \log \pi_i]\right) \tag{4.1}$$

where g_i are samples drawn from the Gumbel$(0,1)$ distribution:

$$g = -\log(-\log(u)), \qquad u \sim \text{Uniform}(0,1) \tag{4.2}$$

In the attention mechanism, the energy between each query and the key is typically transformed to a probability distribution by applying the *softmax* function.

$$\text{softmax} = \frac{\exp(h_i)}{\sum_{j=1}^{k} \exp(h_j)} \tag{4.3}$$

Evidently, the outputs of the *softmax* function will be sparse only if the norm of a single h_i is dominant when compared to other $h_{j \neq i}$, and even then, the outputs will not be discrete but infinitesimally small.

By combining the Gumbel distribution and the softmax function, we obtain the Gumbel-Softmax distribution, where the temperature parameter τ controls the smoothness of the distribution:

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_i) + g_i)/\tau)} \tag{4.4}$$

The closer τ is to zero, the more the Gumbel-Softmax distribution resembles the categorical distribution, while as it approaches $+\infty$, samples become uniform. A visual example of how the value of τ affects the distribution can be seen in Figure 4.2. In our case, the log-probability $\log(\pi_i)$ of each sense representation is the attention energy between the hidden representation and the each sense.

The recommended way of learning models with the Gumbel-Softmax distribution [8] states that we should start with a higher τ and anneal it to 0, as the variance of gradients is large on small temperatures, causing difficulties in the optimization process. As in our case we have to sample discrete values to avoid divergence into the trivial solution, we resort to the Straight-Through [ST; 101] Gumbel Estimator.

The Hard Kumaraswamy Distribution

Using Gumbel-Softmax offers a way to constrain the network to select a single sense for a given hidden representation, but that might not be enough in every case. In this section, we will take on a different perspective – for each sense representation we will make a binary decision whether

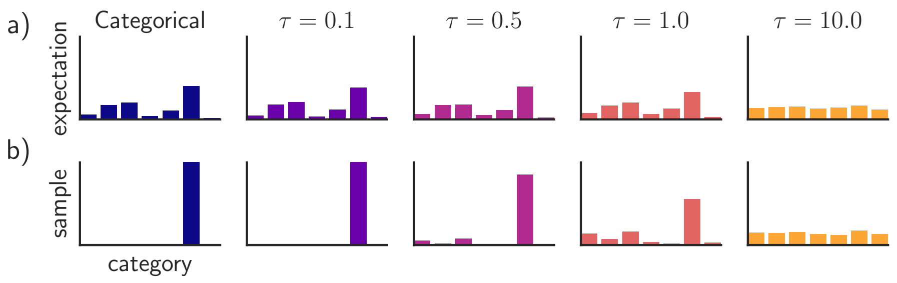


Figure 4.2: Samples from the Gumbel-Softmax distribution range from one-hot encodings (for low temperatures τ) to resembling the uniform distribution (as $\tau \to \infty$). At low temperatures, (a) the expectation of the Gumbel-Softmax random variable resembles the one of a categorical distribution (e.g. the first and second columns). Similarly, (b) samples from the Gumbel-Softmax distribution are identical to categorical ones as $\tau \to 0$. In our case, due to the aforementioned completeness problem, we will only consider $\tau = 0$ throughout our experiments. Figure taken from [8].

that sense should be selected or not, and then constrain the number of selections to not be larger than a predefined maximum K. Introducing such a discrete step in a network would usually require us to resort to the REINFORCE [95] gradient estimation [102], notoriously difficult due to high variance gradient estimates. However, by selecting a convenient probability distribution and exploiting the stretch-and-rectify trick [103], we can obtain a continuous distribution where the probability mass on its edges is nonzero – meaning we seemingly get the *best of both worlds* in form of both differentiability and discreteness.

The authors select the Kumaraswamy distribution [104] due to its similarity to the Beta distribution and for its simple (inverse) cumulative distribution function (cdf, Eq. 4.5), which simplifies inverse transform sampling.

$$F_K^{-1}(u;a,b) = \left(1-(1-u)^{1/b}\right)^{1/a} \tag{4.5}$$

where $u \sim \text{Uniform}(0,1)$. Our next step is *stretching* the support of the distribution to include 0 and 1. For $l < 0$ and $r > 1$, the transformed CDF is as follows:

$$F_T(t;a,b,l,r) = F_K\left((t-l)/(r-l);a,b\right) \tag{4.6}$$

The transformed distribution $T \sim Kuma(a,b,l,r)$ is now defined on the open interval (l,r). Finally, we can rectify our stretched distribution to conflate all the probability mass from an interval to a single point. The authors pass the sample T through a hard-sigmoid function $h = \min(1, \max(0,t))$. The resulting variable is defined over the *closed* interval $[0,1]$, and while the probability of sampling $t = 0$ is equal to 0, the probability of sampling $h = 0$ corresponds to the entire probability mass on the interval $(l,0]$, which can be computed in closed form. The same holds for the interval $[1,r)$ and the probability of sampling $h = 1$. A visual intuition of the process, and the Kumaraswamy distribution before and after stretching, can be seen in Fig. 4.3.

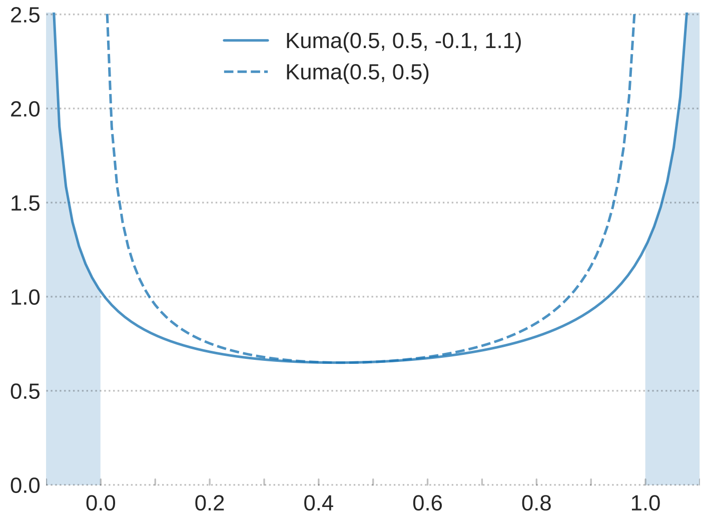


Figure 4.3: The stretched and unstretched variants of the Kumaraswamy distribution. The unstretched variant (dotted blue line) for $a, b = 0.5$ is used as a starting point. The support is then stretched for 0.1 to the interval $l = -0.1, r = 1.1$ (Equation (4.7b)). After applying the hard-sigmoid transformation, the probability mass in the shaded area is conflated to the tail ends of the distribution (Equation (4.7c)). Figure taken from [9].

As the transformed distribution is based on the Kumaraswamy distribution, we can still leverage the reparametrisation trick with a uniform random variable $u \sim \mathrm{Uniform}(0,1)$. The procedure of obtaining samples from the Hard Kumaraswamy distibution would then work as follows: first, we use the use inverse transform sampling to obtain $k \sim F_K$ (Eq. 4.7a). Then we stretch the resulting variable to the (l, r) interval (Eq. 4.7b). Finally, we apply the hard-sigmoid function to obtain a sample from the Hard Kumaraswamy (HardKuma) distribution (Eq. 4.7c).

$$k = F_K^{-1}(u; a, b) \tag{4.7a}$$
$$t = l + (r - l)k \tag{4.7b}$$
$$h = \min(1, \max(0, t)) \,, \tag{4.7c}$$

However, simply using this distribution as part of your neural network does not suffice to induce sparsity as the network has to be incentivized to assign more probability mass to the 0-end of the distribution by appropriately fitting the distribution parameters a and b.

In this part the choice of distribution comes into play again. Due to the simple and tractable CDF of the Kumaraswamy distribution, we have a closed-form expression for the expected L_0

norm of the produced samples over the sense inventory:

$$\begin{aligned} \mathbb{E}_{p(z|x)}[L_0(z)] &\overset{ind}{=} \sum_{i=1}^{K} \mathbb{E}_{p(z_i|x)}[\mathbb{I}[z_i \neq 0]] \\ &= \sum_{i=1}^{K} (1 - \mathbb{P}(Z_i = 0)) \end{aligned} \tag{4.8}$$

Thus, we can add a regularization term as part of our model, which can enforce that the expected number of sampled zeros is close to some predefined value, or in our case, the size of the sense inventory A from which we subtract the number of connections we wish to be nonzero K.

$$\mathbb{L} = \underbrace{\mathrm{CE}(f(x), y)}_{\text{Classifier loss}} + \underbrace{\lambda\,(\mathbb{E}_{p(z|x)}[L_0(z)] - A + K)}_{\text{Sparsity penalty}} \tag{4.9}$$

Vector Quantized Variational AutoEncoders

Perhaps the most natural way of implementing our model draws inspiration from variational autoencoders (VAEs) [105, 106] and vector quantization to introduce discreteness into the latent variable [107]. VAEs, in general, consist of three probabilistic components, each modelled by a neural network: (1) the *prior* distribution over the latent code $p(z)$, (2) an encoder network that parametrizes the *posterior* distribution of the latent variable given the input data $q(z|x)$, and (3) the decoder, with the distribution $p(x|z)$ over the input data based on the latent code. In this case, the latent code presents the autoencoder bottleneck, which incentivizes the model to learn a meaningful representation. Furthermore, due to the inherent randomness in the latent code distribution (as only the parameters of the underlying Gaussian distribution are learned, but the code itself is drawn as a sample) the latent space is *smooth* – meaning that proximity in the latent space carries over to the input space, a feature rarely enforced for deterministic models.

Introducing Discrete Latent Variables. The change introduced in VQ-VAE is in moving away from using Gaussian distributions as the prior and posterior to using the categorical distribution. The VQ-VAE model assumes the existence of a set of latent embeddings, with the categorical distribution acting as an index into the embedding table. The embeddings selected through sampling from the categorical distribution are then passed to the decoder network to reconstruct the original inputs. This setup is the conceptually closest one to our intended model. We will now provide a description of its mathematical framework.

As we mentioned earlier, we assume the existence of a set of latent codes (in our case, a sense inventory) $A \in \mathbb{R}^{A \times d}$, with A being the size of the (discrete) latent space and d the dimensionality of each code (sense representation). As illustrated in Figure 4.4, the model first encodes the input, producing $z_a(x)$. The discrete latent variable is then selected by virtue of a nearest-neighbor lookup:

$$q(z=k|x) = \begin{cases} 1 & \text{for } k = \arg\min_j \|z_a(x) - a_j\|_2 \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

Thus, the input to the decoder after deterministically selecting the nearest latent code is:

$$z_q(x) = a_k, \qquad \text{where } k = \arg\min_j \|z_a(x) - a_j\|_2 \tag{4.11}$$

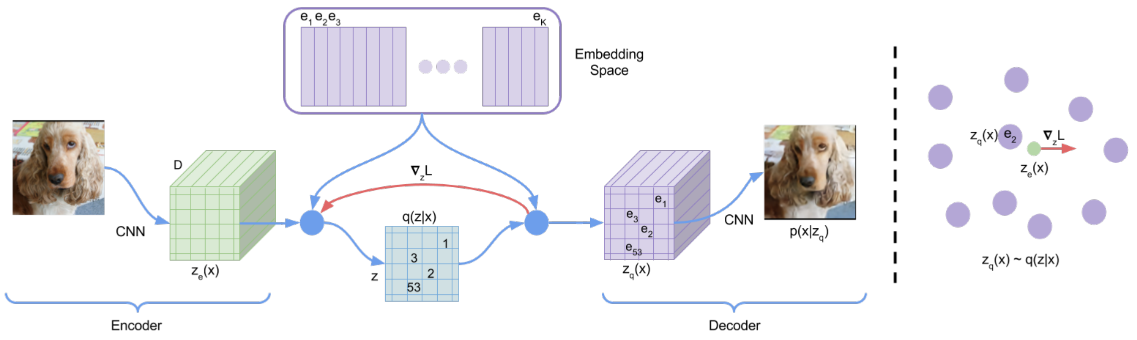


Figure 4.4: The vector-quantized variational autoencoder for images. The input images are first transformed by virtue of a convolutional network, then an appropriate latent embedding is selected through the quantization operation, followed by decoding the input image form the latent code. Figure taken from [98].

By introducing discreteness into the model this way, we still require some method of propagating gradient information through the arg min operator. The approach used in [98] is to simply use the straight-through estimator and pass the gradient over the latent code $\Delta_z\mathbb{L}$ unaltered to the encoder network. As, due to the nearest neighbor search for selecting the latent code, the space of the latent codes and encoder outputs should be shared, the authors hypothesized that the gradient should also contain useful information for the encoder network.

All of the aforementioned methods of inducing sparsity have been used in contexts different than inducing a shared sense inventory. Our task now is to adapt these methods to our autoencoder approach.

4.1.2 Model Variants

Having covered some of the possible methods of incorporating sparsity in neural networks, we will now in detail explain the three main models we will use throughout our experiments, each of them based on one of the sparsity inducing methods. Each of these models can be seen as a variant of the autoencoder from Figure 4.1 with certain components differing based on the sparsity method used. Each model can be summarized through the following set of equations:

$$h_t = \text{encoder}(x_t) \tag{4.12}$$

where h_t are the contextualized representations, while x_t are the input word embeddings. The *encoder* network is either an RNN or a Transformer model. Once we have contextualized the word representations, we feel safe to assume that their senses can be accurately determined by the model. In the next step, we perform sparse selection over the sense inventory. The choice of the sparse operator differs in both the method of sparsity as well as the number of senses selected k.

$$\tilde{a}_t = \frac{1}{k}\sum_{i=1}^{k} a_i, \qquad \text{where } h_t \rightarrow a_i \neq 0 \tag{4.13}$$

Once we have obtained the sequence of sense representations $\{a_t\}$, we perform a decoder pass to reconstruct the original sequence:

$$\hat{x}_t = \text{decoder}(\tilde{a}_t) \tag{4.14}$$

where as the *decoder* we either use an uncontextualized linear transformatinon – to ensure that the input token is reconstructed from the corresponding sense representation in isolation – or a recurrent pointer network [108] to ease the reconstruction process by restricting the decoded tokens to the set of input indices.

Gumbel-Softmax Autoencoder

The first model leverages the Gumbel-Softmax distribution (henceforth GUMBEL) with the temperature parameter τ fixed to zero. We fix the temperature parameter to zero at the cost of gradient information not flowing through the distribution due to the possibility of the model leaving a significant number of nonzero connections active. In practice, we observe that models trained with $\tau \neq 0$ converge to a perfect reconstruction rate very quickly, indicating that this behavior indeed occurs during training. In some cases it is possible to imagine a scenario where the true sense of a word might not be perfectly disambiguated based on context. Also, keeping in mind that we are in no way ensuring that elements from the sense inventory A indeed will correspond to senses (it is indeed imaginable that the model might learn to encode information in such a way that it does not correspond to the sense of a single word, but rather distribute information from encoded sequences over elements from A in some other way which also allows for reconstruction of the input sequence).

To provide some leeway for the model, we also include the multi-head Gumbel-Softmax model (henceforth MH-GUMBEL), which draws inspiration from the multi-head attention mechanism [46] and performs multiple 1-of-k selections over the sense inventory. We do not ensure that the model *has* to select different elements from the inventory with the idea that if a sense is determined correctly, we actually want multiple heads to select the same element. The number of heads (corresponding to the number of items selected from the sense inventory) is a hyperparemeter, which we refer to as the *selection rate*.

Hard Kumaraswamy Autoencoder

When selecting sense representations with the HardKuma distribution (henceforth KUMA), instead of obtaining a categorical sample over the entire sense inventory, we obtain A samples from the binary HardKuma distribution, one for each element of the sense inventory. To sample from the HardKuma distribution, we need to produce its parameters, namely the shape parameters a and b. Following [13], we fix the stretching hyperparameters l and r to -0.1 and 1.1, respectively.

To ensure that the model generates a sparse distribution over the sense inventory, we need to control the expectation of the HardKuma distribution (Equation (4.8)) to produce the intended amount of zeroed out connections, on average. Here we once again use the *selection rate* hyperparameter, although this time it does not determine the number of attention heads but rather the expected number of nonzero samples drawn from the HardKuma distribution. Using the HardKuma distribution as the source of sparsity thus introduces two hyperparameters: (1) the sparsity rate and (2) λ_{kuma}, the regularization strength for the expected number of nonzero values. Note that, in contrast with GUMBEL and VQVAE variants, the KUMA model incorporates a *soft* constraint on the number of selected sense representations.

Vector Quantised Variational Autoencoder

The vector quantised variational autoencoder [98] (henceforth VQVAE) is the most natural approach to our problem, as the authors themselves apply it to a similar task in the vision domain. By applying a nearest-neighbor selection (Equation (4.11)) based on the encoder output h_t, we obtain a single sense representation a_i from the sense inventory.

The authors do note that there is a regularization term important for the model – the *commitment cost*. When selecting representations from the sense inventory, the representation selected is the closest one, however, there is no guarantee of how close these representations are in the learned vector space. To ensure that the sense representation a_i and contextualized representation h_t gravitate towards one another, the authors introduce the following regularization term:

$$||\text{enc}(x) - a||_2^2 \tag{4.15}$$

where in our case $\text{enc}(x)$ is the encoder output sequence $\{h_t\}$, $a = \{a_i\}$ the corresponding sequence of selected sense representations and $||\cdot||$ the squared L_2 norm. This way, through training, a contextualized representation and its selected sense will move closer to each other, ensuring that similar contextualized representations select similar senses from the inventory.

Decoder Variants

We mentioned earlier that we consider two types of decoders (Equation (4.14)) when reconstructing the original tokens based on sense representations. The simpler decoder variant is to leverage a linear layer whose weights are tied to those of the input embeddings [85], effectively performing a nearest-neighbor search in the embedding space and further ensuring that the input space and sense representation space are shared.

However, some models exhibited issues during training and were not able to achieve reasonable reconstruction rates when using a linear decoder. To make the reconstruction task more manageable, we leverage the pointer-generator network [108], which incentivizes the model to select tokens from the input sequence when decoding, a bias useful for tasks such as summarization. The pointer-generator network is a recurrent model that produces a probability distribution over both the output vocabulary as well as the input sequence – which are then averaged to obtain the final probability distribution of the model. In addition, we also incorporate teacher forcing [109] to the recurrent model, which with probability p_{tf} substitutes a generated token with the correct token, thus providing the *correct* input to the next decoding iteration.

Initializing the Sense Inventory

When training our model, we consider two variants of initializing the sense inventory: (1) we randomly initialize the inventory by drawing samples from the Uniform$(-1/A, 1/A)$ distribution; and (2) we perform k-SVD decomposition [94], following the method in [10] and initialize the sense inventory to those values.

Sense Dropout

When running our experiments, we observed that models often fall victim to the *rich get richer* phenomenon, where only a small subset of the sense representations gets tuned, while a large proportion of senses are rarely, if ever, selected. To this end, we introduce a dropout, which acts on the considered set of sense representations when performing sparse selection. Prior to selection, we drop p_{sense_drop} sense representations. We find that this type of regularization greatly helps with the early training phase of models.

4.2 Datasets

Throughout our experiments, we leverage two types of datasets: (1) the pretraining dataset consisting of large amounts of unstructured text and (2) evaluation datasets, which attempt to unvel whether the trained model is successful in learning a useful sense inventory.

4.2.1 Pretraining Dataset

As we are training our model in the autoencoder paradigm, we can use any unstructured source of text as our dataset. In this case, we resort to the English version of the Wikipedia from 2010, provided by the Westbury Lab[*]. We first preprocess the dataset by sentence splitting and then tokenizing it with the SpaCy[†] sentencizer and tokenizer. The tokens are then lowercased and any tokens not containing a single alphanumeric symbol are filtered (e.g., punctuation). The resulting dataset contains 55 million sentences containing 1 billion word tokens, of which 5 million unique ones. The first five lines of the preprocessed dataset are shown as follows:

anarchism
anarchism is a political philosophy which considers the state undesirable unnecessary and harmful and instead promotes a stateless society or anarchy
it seeks to diminish or even abolish authority in the conduct of human relations
anarchists may widely disagree on what additional criteria are required in anarchism
the oxford companion to philosophy says there is no single defining position that all anarchists hold and those considered anarchists at best share a certain family resemblance

When training our model, we select only the most frequent V tokens, with the tokens not part of the vocabulary being replaced with the unknown token `<UNK>`. We further discard all sentences shorter than 5 tokens and tightly pack data batches, concatenating sentences separated with the special `<SEP>` token to maximum size smaller than the maximum allowed sequence length for the model.

4.2.2 Evaluation Datasets

When evaluating our model, we want to use datasets that incorporate sense annotations as part of the evaluation in order to gauge how well has the model captured senses shared between words. The first dataset is the Stanford Contextual Word Similarities (SCWS) dataset, introduced in [38]. Most previous datasets used to evaluate pretrained word embeddings did so

[*] `http://www.psych.ualberta.ca/~westburylab/downloads/westburylab.wikicorp.download.html`
[†] `https://spacy.io/`

without contextual information – thus disregarding the problems of homonymy and polysemy, which might increase or decrease the similarity between words dependent on context.

The SCWS dataset consists of 2003 word pairs and their contextual information from the sentences they occur in. The word pairs contain a total 1712 unique words selected in a way that all categories of low and very polysemous words are included. Each pair of words has been annotated for similarity by ratings from 1 to 10 by 10 different annotators, the scores of which were averaged when performing evaluation. The entirety of the dataset is used as a test set with the model simply providing a representation for the two target words to be used as a basis for cosine similarity. Sample instances from the dataset are shown in Table 4.1.

Similarity	Word 1 context	Word 2 context
4.0	*"…For example , in 1940 , the Imperial **Japanese** Army Air Force bombed Ningbo with ceramic bombs full of fleas … "*	*"…among American women with post-secondary education , African **American** women have made significant advances … "*
8.2	*"…John , preaches to many in **Jerusalem** , and performs many miracles such as healings… "*	*"…majority of the working residents did not work in **Israel** . Over 25 % of the working population … "*
2.6	*"…distributed almost universally across the city every **Wednesday** , and containing news … "*	*"…reality of gang violence by posting **news** stories on the bulletin board… "*

Table 4.1: Sample instances from the SCWS dataset. Each instance consists of an average rating obtained by averaging scores from 10 different annotators and two source sentences. In each source sentence the target words are annotated in **bold**. The goal is to determine the similarity of the target words in context.

Apart from SCWS, we will also evaluate our model on datasets from general downstream applications of neural models, in particular on two sentiment analysis datasets we use throughout our work, SST and IMDb. When evaluating our model on downstream datasets, we remove the decoder part of the autoencoder model and replace it with a new, randomly initialized decoder head which is then tuned for the classification task. The remaining parameters of the model are frozen and used as-is.

4.3 Experimental Details

Although we have considered wider ranges of hyperparameters and model variants throughout our experiments, we only carefully ablated over the ones which contributed the most. Differences between the Transformer and GRU encoders were not significant, which we believe is due both to them being shallow in our implementation, as well as the fact that the sparse selection operator being the bottleneck for the network performance. When initializing sense representations, we use our k-SVD reimplementation based on the MATLAB code from [10]. Full lists of hyperparameters and their considered values shown in Table 4.2.

4.4 Results

In this section, we will detail the experimental results of our models, first by analysing the convergence on the pretraining dataset and then the performance on the contextualized evaluation set and downstream tasks.

4.4.1 Pretraining on Wikipedia

We train each of our models for 5 epochs on the English version of the Wikipedia dataset. Due to relatively long training time, we train each model variant once for each distinct hyperparameter setup. When evaluating how well the model trains on the pretraining task, apart from the reconstruction loss, we are also interested in the *accuracy* of token reconstruction and whether more frequent or less frequent tokens are reconstructed more accurately. While the reasoning for the latter criterion might not be obvious, the main motivating idea is that a common failure case of the model during training is that it learns to primarily reconstruct high frequency tokens and disregard the rest. We track the following metrics during training: (1) the average frequency rank (AFR) of words (higher frequent tokens are assigned lower indices in the vocabulary); (2) the average predicted frequency rank (APR) of words; and (3) the average rank of correctly reconstructed (ACR) words. What we want to observe in a model that is training properly is the values of these three metrics in close proximity to one another as the training progresses. If the average predicted frequency rank is significantly higher than the average frequency rank, then our model is assigning too much value to reconstructing rare words, and if the average predicted frequency rank is signicantly lower than the average frequency rank, our model is reconstructing only high frequency tokens. In Figure 4.5 we present the plots of predicted, correctly predicted and actual word ranks throughout training. We immediately notice that the KUMA and GUMBEL models struggle with representing lower frequency words and resort to predicting more frequent tokens.

4.4.2 Performance on Contextualized Similarity

The results of our models as well as models from related work are shown in Table 4.3. The models are scored with respect to the Spearman correlation coefficient with the average similarity rating of human annotators on the SCWS dataset. The results demonstrate our models falling short of performance of competing models for learning sense representations, which we attribute to the instability of training models with such a large degree of sparsity.

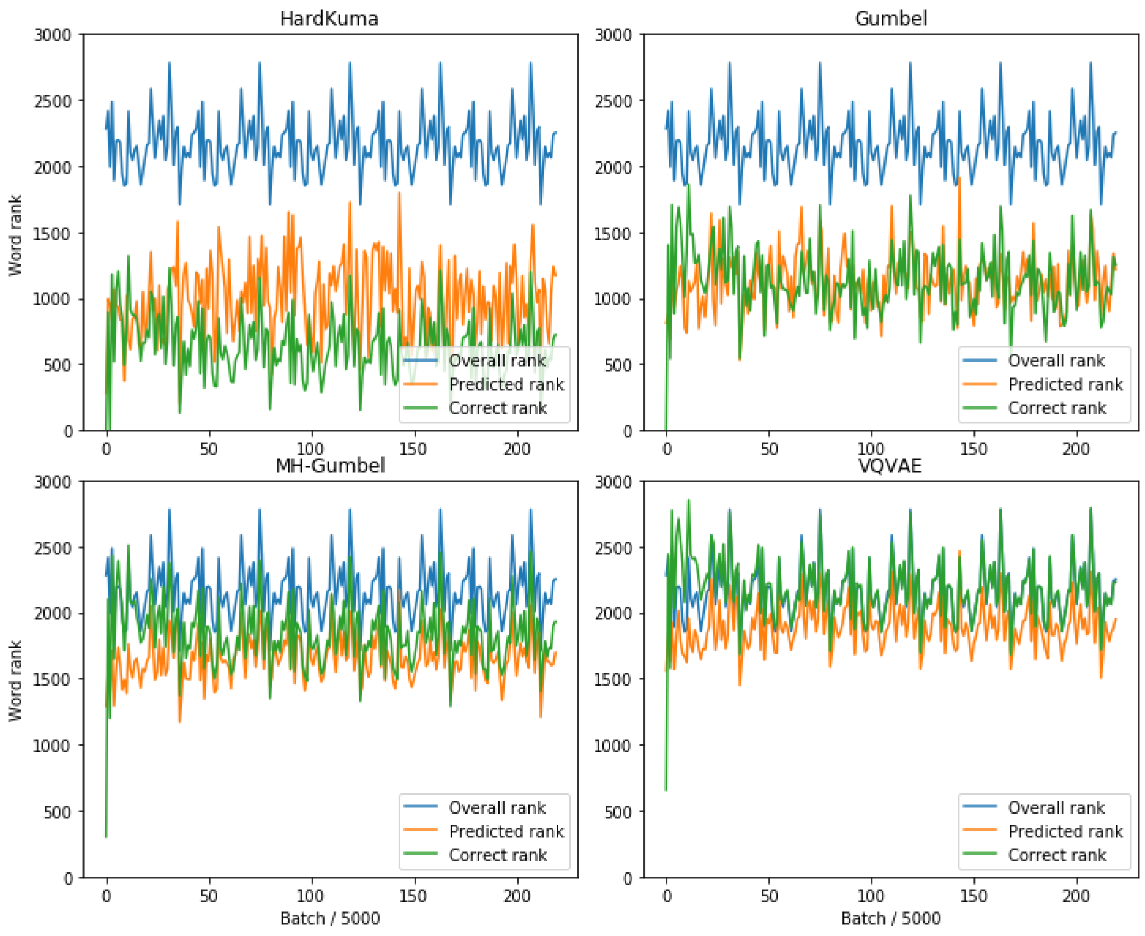


Figure 4.5: Predicted word rank statistics when pretraining on the Wikipedia dataset. More frequent words have a lower vocabulary rank, while rare words have a high vocabulary rank. Ideally, we would aim that the correct and predicted ranks overlap. The averages are computed over 5000 batches.

.

Nevertheless, we believe there is still room for improvement as most of the models show slow but positive trends when training. In Figure 4.6 we plot the Spearman correlation coefficient on the SCWS dataset evaluated every 5000 batches. We only plot models where the sense representations are initialized by k-SVD as those perform significantly better. We can see that the KUMA and GUMBEL models perform worse, while the MH-GUMBEL and VQVAE models possess an upward trajectory while training, although also exhibiting high variance in correlation scores.

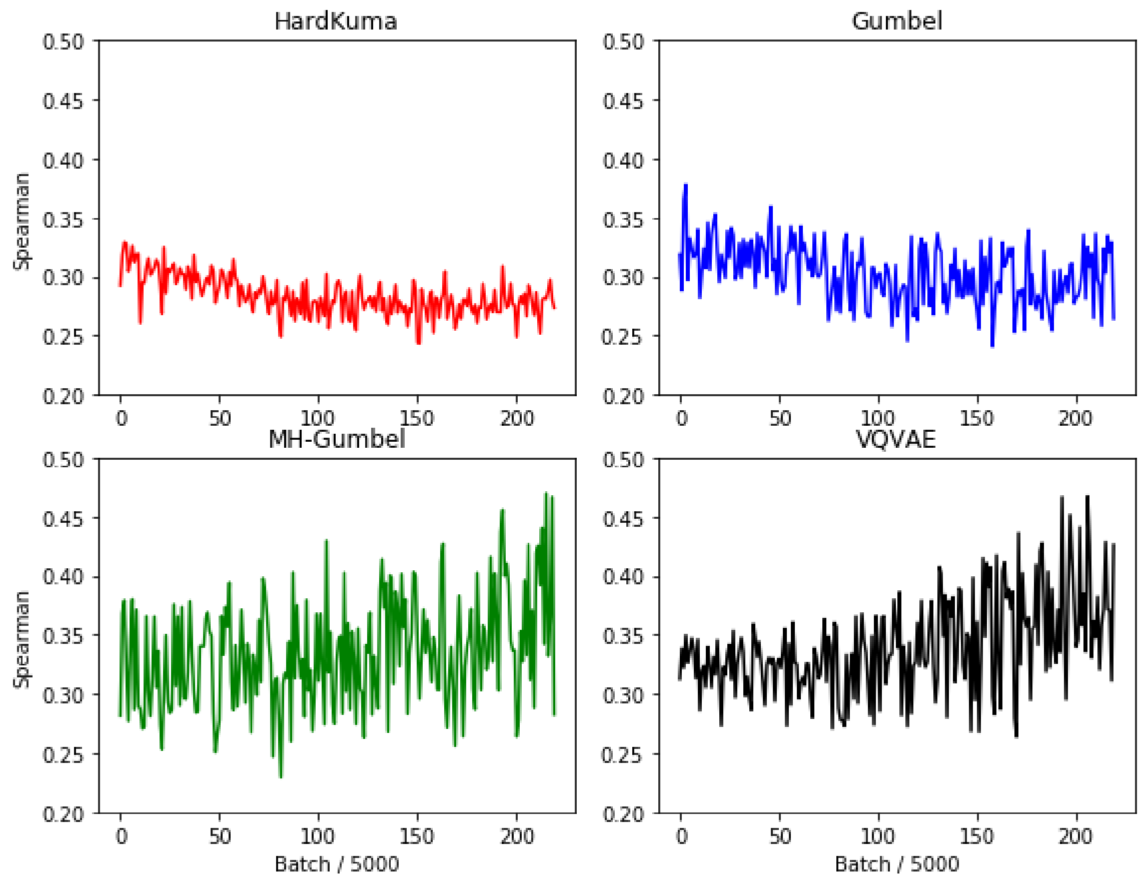


Figure 4.6: Spearman correlation coefficient $\rho \times 100$ on the SCWS dataset for all model variants with k-SVD sense initialization. The correlation coefficient is evaluated after every 5000 pretraining batches.

.

4.4.3 Performance on Downstream Tasks

In order to gauge whether the learned sense representations are useful in downstream tasks, we evaluate the encoding component of our models on the SST and IMDb datasets. We leverage the encoding component of our models by using sense representations obtained from Equation (4.13) for each model. The sequence of sense representations is then treated as input to a LSTM + self-attention model (Figure 2.5). We train each classification model for 5 epochs and evaluate the best performing model with respect to accuracy on the validation split. During training, the sense encoding component is not optimized. We compare our models to the standard variant which leverages GloVe [28] embeddings. We only report results of models where the sense encodings are initialized with representations obtained through k-SVD due to their consistently higher performance.

In this chapter, we have analysed the feasibility of training an autoencoder-based approach to learning shared sense representations. We have shown that the considered approaches of inducing sparsity in neural models suffer from difficulties with processing a high level of sparsity, but also that it is possible to some extent to learn valuable representations. The main conclusions we draw from the presented results are that the stability of training sparse models is very volatile, and that the early stage of training is key for convergence. We believe this line of work to be worth pursuing for both the implications in cross-lingual transfer of the sense representations, which could greatly benefit low-resource languages, and the natural interpretability of having explicit sense representations as part of the trained model.

General hyperparameters	
Embedding dim	300
Freeze embeddings	$\{\text{True}, \text{False}\}$
Tie embeddings	$\{\text{True}, \text{False}\}$
Encoder hidden dim	150
Encoder type	$\{\text{GRU}, \text{Transformer}\}$
Encoder layers	2
Learning rate	$e-4$
Num epochs	5
Batch size	32
Sense inventory size	$\{2000, 3000\}$
Sense dim	300
Sense dropout	0.3
Initialize senses	$\{\text{True}, \text{False}\}$
Pointer network	$\{\text{True}, \text{False}\}$
Pointer network blending	0.5
Teacher forcing	1.0
Sparse selection hyperparameters	
Selection type	$\{\text{Kuma}, \text{Gumbel}, \text{MH-Gumbel}, \text{VQVAE}\}$
Selection sparsity	5 (Kuma), 2 (MH-Gumbel)
Commitment cost (VQVAE)	$\{0.25, 1., 2.\}$

Table 4.2: Considered hyperparameters of models throughout experiments. The blending factor and teacher forcing are used only together with the autoregressive pointer network decoding.

Method	Spearman Coefficient
MSSG [42]	0.567
GloVe [28]	0.573
Skip-gram [4]	0.622
k-SVD [10]	**0.652**
HardKuma	0.273
Gumbel	0.281
HardKuma & init	0.329
MH-Gumbel	0.356
Gumbel & init	0.377
VQVAE	0.422
VQVAE & init	0.467
MH-2-Gumbel & init	0.470

Table 4.3: The Spearman $\rho \times 100$ coefficient of the similarity of encoded word representations with respect to human annonations on the SCWS dataset. Best performing model highlighted in **bold**.

Model	SST	IMDb
GloVe [28]	80.34	**86.24**
Kuma	50.00	53.12
Gumbel	55.21	57.36
MH-Gumbel	77.31	80.15
VQVAE	**81.35**	84.5

Table 4.4: Classification accuracy on the test sets portions of the SST and IMDb datasets. Each model is trained for 5 epochs, and the best performing model on the validation set is then evaluated on the test set. Best results highlighted in **bold**.

Chapter 5

The Iterative Recursive Attention Mechanism

Recurrent neural networks process data sequentially, a paradigm that does not necessarily fit the hierarchical, phrase-based syntactic structure of sentences. When understanding the final decision of a model, it would be beneficial for any retrospective explanation model if the decision could be decomposed as a series of inferences steps, each of which can swing the decision either way. Inspired by work of [110, 111], we proposed the iterative recursive attention model [IRAM; 112], which recursively constructs representations of the input sequence – and in contrast with previous work, the intermediately constructed representations can be reused in further iterations. Such unfolding of the inference process into a series of interpretable steps is aimed to both treat contrastive clauses and negations as well as allow users to better determine the shortcomings of their models.

Apart from the obvious benefits in view of model interpretability through observing the distinct inference steps, IRAM simplifies the task of the attention mechanism by removing the requirement of reaching a prediction in a single attention step. The simplification allows the model to isolate the contrastive aspects of the sentiment analysis datasets we considered, semantically composing both the positive and negative polarity aspects of data and finally weighing them to produce the final representation.

5.1 Methodology

The goal of our modifications to the attention mechanism is to decompose the inference process into a series of steps. We do this by extending the standard way of applying the self-attention mechanism – instead of passing the result of a single self-attention step to the classifier network, we first transform it and then concatenate it to the original sequence. By using multiple attention steps, the model is able to construct a recursive representation and process the input sequence bit by bit. The upshot of this approach is that apart from simplifying the task for the model, we also gain the ability to inspect how the model weighs different parts of the sentence and then combines them to obtain the final decision.

Our model can be viewed as carrying out three consecutive phases: (1) the *encoding phase*, where the word representations are contextualized by an encoder network, (2) the *attention phase*, which leverages recursive attention to isolate and combine different subsets of the input sequence, and (3) the *classification phase*, where the final representation is fed to the classifier. A visual representation of the model is given in Figure 5.1.

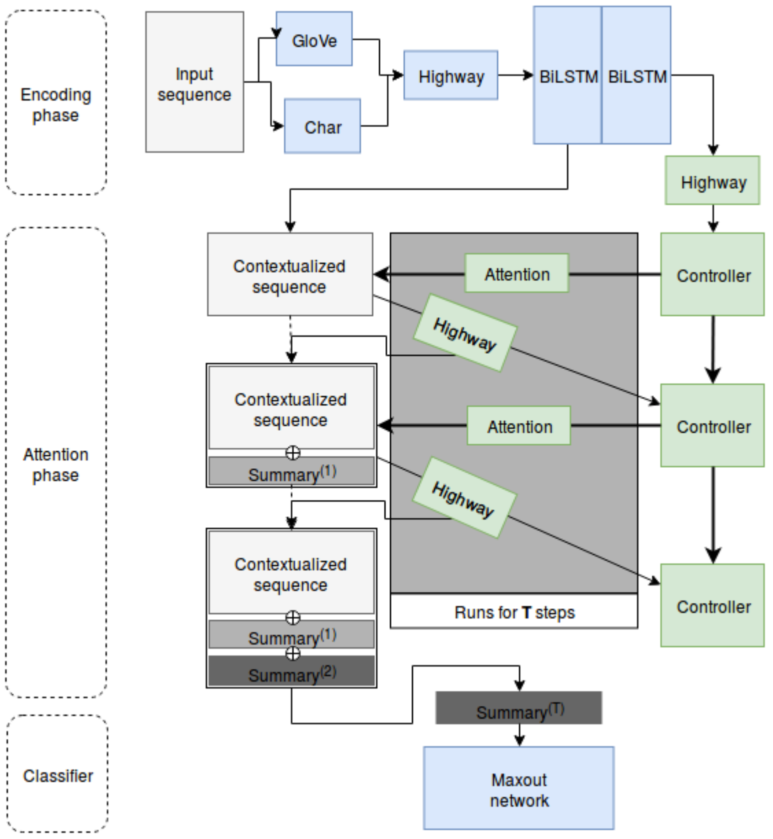


Figure 5.1: The iterative recursive attention model (IRAM). Green-colored components share their parameters with components of the same type. Highlighted in gray is one iteration of IRAM.

5.1.1 The Encoding Phase

In the encoding phase, we leverage both GloVe word embeddings [28] and character n-gram embeddings [113]. All character n-gram embeddings for an input word are averaged over and the word and character representations are then concatenated in a single vector. We follow the procedure from [114] and apply a transformation to the concatenated representations. However, we deviate from their procedure and use a highway network [80], which we found to perform better when compared to a ReLU feedforward network. We then take the transformed, but uncontextualized representations and contextualize them by means of a bidirectional LSTM network. We conceptually split the LSTM network into two parts: the lower l_{ctx} layers are used to contextualize the word representations, while the uppermost l_{query} layers are used to produce a fixed-size representation of the now-contextualized sequence, then used as the initial query for the recursive attention mechanism. Intuitively, the split should incite a division of labor between the two parts of the network: contextualization network only has to memorize the local information specific to each word (e.g., verb tense, noun gender) in order to transform its representation, while comprehension network needs to model aspects of meaning pertaining to the entire sequence (e.g., the overall sentiment of the sentence, locations of sentiment-bearing phrases).

We use a single $(l_{ctx} + l_{query})$-layered BiLSTM, however the contextualized word representations h_t are obtained as the outputs of the l_{ctx}-th layer, while the cell state from the last timestep of the last layer is used as the sequence representation $\hat{x}$. This final representation is also passed through a highway network to tune the sequence representation into an initial query.

5.1.2 The Attention Phase

In the attention phase (Figure 5.2), our inputs are the initial query ($q^{(0)}$) and a sequence of hidden states $H = \{h_i\}_1^N$ obtained from the encoding phase. The recursive attention mechanism uses a gated recurrent network [75] as the *controller*, which updates the query representation with respect to the output of the attention step between subsequent iterations.

In each step t of the iterative attention mechanism we attend over the hidden state sequence H with respect to the current query $q^{(t)}$. For the attention mechanism, we use bilinear attention [115].

$$a = softmax(q^{(t)WH}) \tag{5.1}$$

where W is the parameter vector of the bilinear attention mechanism, and a the attention distribution over the hidden representation. The attention weights are then used to compute the summary in step t as a convex combination of hidden representations:

$$\hat{s}^{(t)} = \sum_i^N a_i^{(t)} h_i \tag{5.2}$$

Since we intend to use $\hat{s}^{(t)}$ as a hidden representation in the next iteration of the attention mechanism, we need to nonlinearly transform it so the new representation is no longer linearly dependent on the remaining hidden representations. To do this, we again leverage the highway network to transform the initial summary, $s^{(t)} = \text{Highway}(\hat{s}^{(t)})$. The output of the highway network is then added to the set of hidden representations $H = \{h_i, \dots, h_N, s^{(1)}, \dots, s^{(T)}\}$.

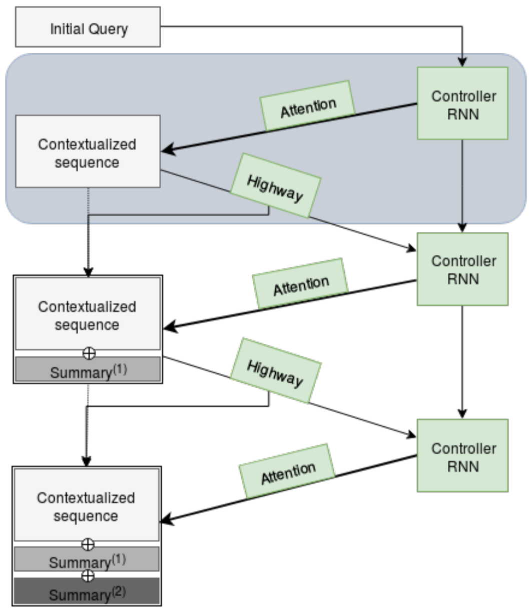


Figure 5.2: The attention component of the iterative recursive attention model. Green-colored components share their parameters with components of the same type. Highlighted in gray is one iteration of IRAM.

5.1.3 The Classification Phase

Lastly, in the classification phase the sequence representation $s^{(T)}$ is fed into a maxout network [116] to obtain the class-conditional probabilities.

5.1.4 Regularizing Attention

Our goal was for the model to focus on different task-related aspects of input data in each iteration of recursive attention. However, if the model is in no way incentivized to propagate information through summaries, we find that often it converges to a solution where it attends on the same set of input representations in each step. To prevent this unwanted behavior from happening, and incentivize the model to focus on different aspects of input data in every step, we regularize it by minimizing the pairwise dot product of the attention distribution between all attention iterations:

$$L_{attn} = \frac{\gamma}{2T} \sum_{i \neq j} [AA^T]_{ij} \tag{5.3}$$

where γ is the regularization weight hyperparameter abd $A \in R^{T\times(N+T-1)}$ is a matrix containing the attention distribution generated in each of the T steps over the initial N inputs and the T added summaries. Note that the final summary is not part of the regularization term as it cannot be attended over, so the final number of columns is $N+T-1$.

While this regularization penalty does incentivize the model to focus on different elements of the input sequence in different recursive attention iterations, there is still a trivial way for the model to bypass the penalty where the model performs an attention step over the inputs only in the first iteration, and simply places the entire attention mass on the previous summary in the next iteration. However, we find that the trained model resorts to this behavior only in case the input instances are straightforward and do not contain contrastive segments. We will further illustrate this behavior in Section 5.4.2.

5.2 Datasets

We evaluate our model on two sentiment analysis datasets also used in Section 3.2, SST and IMDb. Along with the binary version of SST, here we also include the fine-grained classification task where classes range from 1, indicating very negative, to 5, indicating very positive. When preprocessing the datasets, we use SpaCy[*] to tokenize IMDb, we lowercase all tokens and remove tokens which do not contain a single alphanumeric character. We truncate each sentence from the datasets to a maximum of 200 tokens.

5.3 Experimental Details

Throughout our experiments we have evaluated a wide range of hyperparameters by manual tuning, however only the ones deemed most important were evaluated exhaustively. The full list of the hyperparameter search space of our model can be seen in Table 5.1.

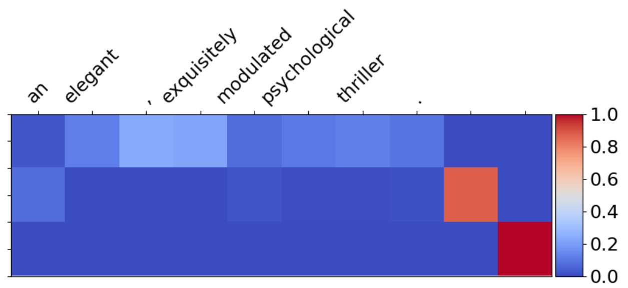
5.4 Results

In this section, we will report the experimental results of our model on the aforementioned datasets and evaluate how much distinct components of the model contribute to its performance. Through manual hyperparameter tuning on the validation splits of the datasets, we have determined that $T = 3$ steps of the iterative recursive attention mechanism and attention regularization weight of $\gamma = 0.0003$ are the best values of the considered hyperparameters. In Table 5.2 we report the accuracy scores of our best models on the test portions of the SST and IMDb datasets along with competing models at time of publication.

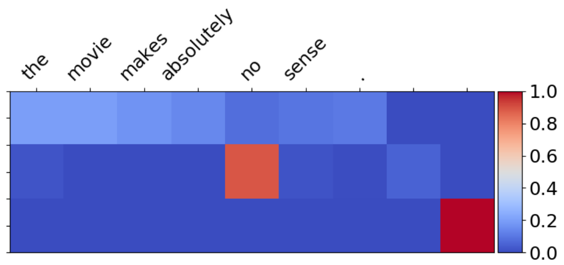
5.4.1 Ablation Experiments

Through adding various complex components to our model we introduced a number of confounders, which might cloud the relevance of the iterative attention mechanism itself. In order to determine the effect of each of the added components on the overall score, we evaluate the

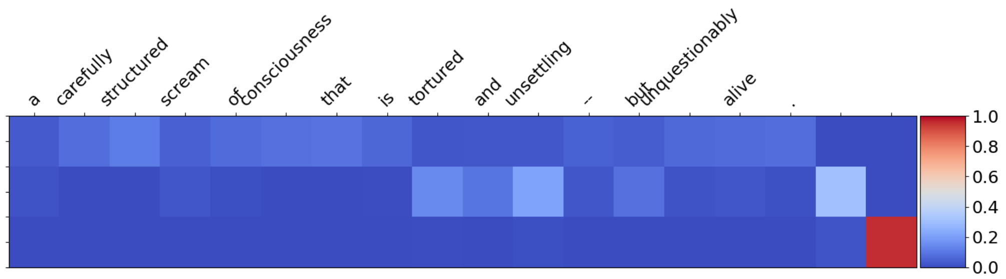
[*] `https://spacy.io/`



(a) A simple unipolar sentence



(b) A sentence with a negation



(c) A contrastive multipolar sentence

Figure 5.3: Visualization of attention across sentence words (horizontal) and T=3 time steps (vertical). The last T-1 columns contain the attention weights over the result of the previous attentive query.

performance of the full model on the binary SST dataset with the remaining hyperparameters fixed and each component removed in isolation. We also report the results of our *vanilla model*, where we omit all of the removed components at the same time. Results are given in Table 5.3

5.4.2 Visualizing the Recursive Approach

To gain an intuition about the working of IRAM, we visually analyzed its attention mechanism on a number of sentences from our dataset. We limit ourselves to examples from the test set of the SST dataset as the length of examples is manageable for visualization. We isolate three specific cases where the attention mechanism demonstrates interesting results: (1) simple unipolar sentences, (2) sentences with negations, and (3) multipolar sentences. Figure 5.3 gives examples for these types of sentences.

In the most straightforward example, the unipolar case (Figure 5.3a), we can recognize an instance where the model does not require multiple iterations as the instance contains no contrastive elements or polarity flipping. In that case, the network simply learns to propagate information between iterations of attention, evident from most of the attention mass in the second and third rows (steps) being attributed to the representation from the previous iteration.

More interesting cases involve negations and multipolar sentences. In Figure 5.3b we see how the model handles negation in a sentence. Attention is first distributed over all words except the negator. Then, in the second iteration, the mechanism combines the previously learned representation with the negation. We interpret this behavior as flipping the polarity of the representation – solely recognizing a negator and additively summing its representation, as is traditionally done when applying vanilla attention, would be unreasonably difficult. Due to its recursive nature, our model provides a more natural way of processing such sequences, by first constructing a representation of the otherwise positive expression, and then negating it.

Finally, in Figure 5.3c we show a contrastive multipolar sentence, where in each iteration the model focuses on subsets of tokens with different polarity before combining their representations. We believe that the iterative nature of this approach allows the model a simpler way of *weighing* the contrastive elements of the input sequence when compared to simultaneously taking everything into account.

We have demonstrated that building recursive representations of sentences not only benefits interpretabity of models through inspecting attention weights in each recursive step, but also improves model performance on sentiment analysis tasks. The performance improvement is something we believe is caused by the improved semantic composition capability of the introduced model, where the otherwise complex functional dependence of phrases in the sentence structure can now be processed in multiple iterations instead of a single attentive step. We believe that similar modifications which allow the model to process complex instances in multiple steps can only benefit both the capabilities of neural models and broaden the insights that can be drawn about the inner workings of models by practitioners.

Model hyperparameters	
Word embedding dim	300
Char embedding dim	100
Embedding dropout	0.1
Freeze embeddings	$\{\text{True}, \text{False}\}$
RNN cell	LSTM
RNN hidden dim	$\{400, 500, 1000\}$
Contextualization layers	2
Query tuning layers	1
Optimizer	Adam + AmsGrad
Weight decay	$3e{-}5$
AmsGrad α	$3e{-}4$
Learning rate	$1e{-}3$
Grad. clipping	1
Batch size	$\{32, 64\}$
Num Epochs	5
Dropout	$\{0.1, 0.2, 0.3, 0.4\}$
Maxout network dim	200
Maxout network layers	2
Maxout network pool	4
Attention regularization γ	$3e{-}4$
Recursive steps T	$\{1, 2, 3, 4, 5\}$

Table 5.1: Considered hyperparameters for IRAM throughout experiments. As the hidden state size of the recurrent network was increased, the dropout had to be increased accordingly for stable convergence. As long as attention regularization was set to a low enough value, minor differences did not affect model traning nor performance.

SST		SST-5		IMDb	
NSE [117]	89.7	NSE [117]	52.8	IRAM	91.2
IRAM	90.1	IRAM	53.7	TRNN [118]	93.8
BCN + CoVe [114]	90.3	BCN + CoVe [114]	53.7	oh-LSTM [119]	**94.1**
bmLSTM [120]	**91.8**	BCN + ELMo [6]	**54.7**	Virtual [121]	**94.1**

Table 5.2: Classification accuracy on the test sets portions of the considered datasets. Our model underlined, best results highlighted in **bold**. Note that most of the competing models leverage pretrained representations from large language models.

Removed component	**Accuracy**
Full model	**90.1**
Vanilla model	88.7
– char n-grams	89.3
– query fine-tune	89.8
– embedding fine-tune	89.3

Table 5.3: Effect of removing introduced components on performance of the IRAM. The vanilla model presents the lower bound as all of the ablated components are removed in it. Results reported are averages over 3 runs.

Chapter 6

Discussion

After this brief journey into the inner workings of the recurrent neural network algorithm for processing natural language, we are left with some novel insights and some troubling questions.

RNNs or Transformers. We have demonstrated that RNN variants, including GRUs and LSTMs, have underlying issues that likely cause convergence problems on larger depths in terms of number of stacked RNN layers. Issues of lateral information leakage and vertical information blockage, while to an extent solved through word-level regularization in our work, are still not completely eliminated. The need for tuning a regularization term in the era of deep models and long training times is especially time-consuming, and when a model has its own additional regularization penalties, optimizing multiple moving targets might be a computationally infeasible task to solve. To this end, we should aim for a *yes* or *no* answer to whether a model has resolved these issues – ideally by virtue of introducing a novel model family where the issues of lateral information leakage and vertical information blockage are eliminated altogether. From this perspective, the usage of Transformer networks, which due to their residual connections naturally do not suffer from vertical information blockage, seems as a better option. However Transformer networks also exhibit problems of their own in the often detrimental lack of a state [122], issues with representing long sequences due to the square complexity of the attention mechanism [123], and the fact that they often disregard word order [124].

Usefulness of Interpretability. A point of contention common in machine learning is whether preference should be given to interpretable models at, perhaps, the price of performance. While for high-stakes decision scenarios this question has a definitive answer, if humans are not part of the decision equation, why should we care about the interpretability of models? "All models are wrong, but some models are useful", state [125], thus the question becomes what is considered *useful* for a practitioner. One of the main arguments for the usefulness of interpretability is determining the blind spots of neural models, refering to either detecting artifacts in training data [53] or finding [12] a major issue in a popular model family through a symptom in its explainability. Thus, we believe that focus should be placed towards making models interpretable rather than not, even at the price of downstream performance.

Modeling Senses Explicitly. One line of work we tackled was that of learning shared sense representations by inducing sparsity in deep networks. While the model reached reasonable performance, it did not outperform existing methods of learning single-prototype representations on downstream tasks to warrant the increase in training time and number of parameters. Implications of learning an explicit sense inventory are far reaching, as senses are naturally independent of the source language, thus introducing the possibility of a sense inventory learned on a resource-rich language being transferred to a low-resource one. Apart from cross-lingual transfer, an obvious benefit in making sense information explicit is understanding the inner workings of models – sense representations are inherently interpretable and simple to label by nearest neighbor search in the input embedding space [10]. Thus, we can relatively easily know when a concept is not recognized by a network by investigating the sense encodings for a particular input sequence. However, such an algorithm requires sparsity in the sense selection step, and the sparsity rate required has shown to be far too large to ensure stable convergence of such algorithms. We believe this line of work to still be worth pursuing as sparsity methods and their respective optimization techniques evolve as such models could provide benefit the field of NLP, especially for low resource languages.

Chapter 7

Conclusion

Recent advancements in natural language processing (NLP) have seen the research community move away from recurrent neural networks towards predominantly using variants of the Transformer model. However, it is my belief that no conclusion can be made with respect to which of the models is the better choice long-term. The standard variant of the Transformer model is undoubtedly better suited for pretraining in the large language model (LLM) paradigm due to better depth scaling, but fully attentional networks must resort to suboptimal solutions of representing positions in form of positional encoding due to inherent statelessness. The optimal model for processing and understanding language is unlikely to yet to have been discovered, and there is merit in analysing, and possibly improving, recurrent networks to obtain insights as to what makes them superior to Transformer networks in aspects such as training from scratch for classification tasks – insights that can in the future be used to improve either of the two competing algorithms.

In the scope of this doctoral thesis, we have first delineated two problems of recurrent models in lateral information leakage and vertical information blockage. These issues had symptoms that reached far and wide into work on interpreting neural models but had a simple cause – the recurrent state passed laterally through the network often dominated information originating from inputs at each timestep. By incorporating token-level regularization into the network we have demonstrated that the network is able to keep the best of both worlds by retaining downstream task performance as well as enhancing faithfulness of model intepretability. We reported consistent results across a number of single-sequence classification tasks corroborating our hypotheses and hopefully paving the way towards attention being used as a method of model interpretability.

We have further attempted to incorporate recent advances in inducing sparsity in neural networks to create a model for learning shared word representations. The model leveraged the fact that multiple surface forms often map to the same sense in order to improve the downside of previous multi-prototype representation algorithms, where rare senses were often starved for contextual information and thus learned poorly. However, the most promising algorithms in Gumbel-Softmax, Hard Kumaraswamy, and VQVAE have been difficult to incorporate at the required level of sparsity.

Finally, we have presented a novel iterative recursive attention algorithm that biases the model to learn a more interpretable and simpler way of combining contrastive information in natural language data. The model has demonstrated competetive performance on standardized sentiment analysis datasets, indicating that further similar improvements might bridge the gap between stateful recurrent models and purely attentional networks.

Taken together, our lines of research paint a larger picture – a one where existing models can be significantly improved upon if the causes of known symptoms are discovered. By leveraging interpretability methods it is not only easier for laymen to understand the decision process of the otherwise black-box models but also for practitioners to understand which aspects of input data are not encoded well and what the blind spots of the model are. Interpretability in recurrent models, in both explicit encoding of sense information as well as semantic composition, is the overarching goal of this thesis, and one we believe we have made significant progress towards. Taking everything into consideration, we outline major avenues of future work below.

Information propagation in recurrent networks. The issues detected in Chapter 3 have highlighted that recurrent networks converge to a dominantly lateral flow of information. We believe this is the main cause of diminishing returns for deeper recurrent networks, and by enforcing vertical information propagation through word-level regularization, training of deeper recurrent models could not only be possible but also compete with deep Transformer networks. In follow-up research, word-level regularization has shown not only to improve faithfulness of interpretability for attention-based explainability methods, but also for saliency approaches, highlighting that the underlying issue was indeed significant.

Learning Shared Sense Representations. In work covered in Chapter 4 we have demonstrated that, while possible, it might not yet be feasible to learn shared sense representations with current methods of inducing sparsity in neural networks. When only a small proportion of representations is used, the model often suffers from the *rich get richer* phenomenon, where a small set of sense representations selected (and tuned) initially then become selected even in contexts where they are not a good fit, simply due to them being exposed to more domain data quicker. To resolve this problem, it is possible to either (1) opt for better initialization techniques so that more initial sense representations are a good fit and thus will not be ingored by the model; (2) incorporate additional regularization where, at least in the initial stages of training, the frequency of selection of each sense representation has to be uniform across the inventory to ensure all senses are tuned; (3) perform a hierarchical selection process where the representations are selected in multiple levels, thus reducing the sparsity rate in each step and keeping the sense inventory size fixed at the cost of multiple discrete choices.

Building recursive representations. The model we proposed and outlined in Chapter 5 has shown that incorporating inductive biases, even complex ones, can have positive effects on both model interpretability and performance. We believe there is merit in studying the way neural model construct internal representations in order to determine which linguistic phenomena are captured well in the semantic composition process of the neural network, which can in turn help practitioners incorporate inductive biases which can simplify the learning process.

Bibliography

[1]Harris, Z. S., “Distributional structure”, Word, Vol. 10, No. 2-3, 1954, str. 146–162.

[2]Firth, J. R., “A synopsis of linguistic theory, 1930-1955”, Studies in linguistic analysis, 1957.

[3]Schutze, H., “Ambiguity in language learning: computational and cognitive models”, Doktorski rad, Stanford University, 1995.

[4]Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., Dean, J., “Distributed representations of words and phrases and their compositionality”, in Advances in neural information processing systems, 2013, str. 3111–3119.

[5]Li, J., Jurafsky, D., “Do multi-sense embeddings improve natural language understanding?”, arXiv preprint arXiv:1506.01070, 2015.

[6]Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L., “Deep contextualized word representations”, in Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, str. 2227–2237, dostupno na: https://aclanthology.org/N18-1202

[7]Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., “Bert: Pre-training of deep bidirectional transformers for language understanding”, in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, str. 4171–4186.

[8]Jang, E., Gu, S., Poole, B., “Categorical reparameterization with gumbel-softmax”, arXiv preprint arXiv:1611.01144, 2016.

[9]Bastings, J., Aziz, W., Titov, I., “Interpretable neural predictions with differentiable binary variables”, arXiv preprint arXiv:1905.08160, 2019.

[10]Arora, S., Li, Y., Liang, Y., Ma, T., Risteski, A., “Linear algebraic structure of word senses, with applications to polysemy”, Transactions of the Association for Computational Linguistics, Vol. 6, 2018, str. 483–495.

[11]Bahdanau, D., Cho, K., Bengio, Y., “Neural machine translation by jointly learning to align and translate”, arXiv preprint arXiv:1409.0473, 2014.

[12]Jain, S., Wallace, B. C., “Attention is not explanation”, arXiv preprint arXiv:1902.10186, 2019.

[13]Bastings, J., Filippova, K., “The elephant in the interpretability room: Why use attention as explanation when we have saliency methods?”, arXiv preprint arXiv:2010.05607, 2020.

[14]Manning, C., Schutze, H., Foundations of statistical natural language processing. MIT press, 1999.

[15]Salton, G., Wong, A., Yang, C.-S., “A vector space model for automatic indexing”, Communications of the ACM, Vol. 18, No. 11, 1975, str. 613–620.

[16]Turney, P. D., Pantel, P., “From frequency to meaning: Vector space models of semantics”, Journal of artificial intelligence research, Vol. 37, 2010, str. 141–188.

[17]Rapp, R., “Word sense discovery based on sense descriptor dissimilarity”, in Proceedings of Machine Translation Summit IX: Papers, New Orleans, USA, Sep. 23-27 2003, dostupno na: https://aclanthology.org/2003.mtsummit-papers.42

[18]Linnainmaa, S., “The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors”, Doktorski rad, Master’s Thesis (in Finnish), Univ. Helsinki, 1970.

[19]Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C., “A neural probabilistic language model”, Journal of machine learning research, Vol. 3, No. Feb, 2003, str. 1137–1155.

[20]Mnih, A., Hinton, G., “Three new graphical models for statistical language modelling”, in Proceedings of the 24th international conference on Machine learning, 2007, str. 641–648.

[21]Turian, J., Ratinov, L.-A., Bengio, Y., “Word representations: A simple and general method for semi-supervised learning”, in Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. Uppsala, Sweden: Association for Computational Linguistics, Jul. 2010, str. 384–394, dostupno na: https://aclanthology.org/P10-1040

[22]Mnih, A., Hinton, G. E., “A scalable hierarchical distributed language model”, in Advances in Neural Information Processing Systems, Koller, D., Schuurmans, D., Bengio, Y., Bottou, L., (ur.), Vol. 21. Curran Associates, Inc., 2008, dostupno na: https://proceedings.neurips.cc/paper/2008/file/1e056d2b0ebd5c878c550da6ac5d3724-Paper.pdf

[23]Mikolov, T., Karafiát, M., Burget, L., Cernock ỳ, J., Khudanpur, S., “Recurrent neural network based language model.”, in Interspeech, Vol. 2, No. 3. Makuhari, 2010, str. 1045–1048.

[24]Collobert, R., Weston, J., “A unified architecture for natural language processing: Deep neural networks with multitask learning”, in Proceedings of the 25th international conference on Machine learning, 2008, str. 160–167.

[25]Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K. J., “Phoneme recognition using time-delay neural networks”, IEEE transactions on acoustics, speech, and signal processing, Vol. 37, No. 3, 1989, str. 328–339.

[26]Mikolov, T., Chen, K., Corrado, G., Dean, J., “Efficient estimation of word representations in vector space”, arXiv preprint arXiv:1301.3781, 2013.

[27]Mnih, A., Kavukcuoglu, K., “Learning word embeddings efficiently with noise-contrastive estimation”, Advances in neural information processing systems, Vol. 26, 2013.

[28]Pennington, J., Socher, R., Manning, C. D., “Glove: Global vectors for word representation”, in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, str. 1532–1543.

[29]Arora, S., Liang, Y., Ma, T., “A simple but tough-to-beat baseline for sentence embeddings”, in International conference on learning representations, 2017.

[30]Levy, O., Goldberg, Y., Dagan, I., “Improving distributional similarity with lessons learned from word embeddings”, Transactions of the association for computational linguistics, Vol. 3, 2015, str. 211–225.

[31]Levy, O., Goldberg, Y., “Neural word embedding as implicit matrix factorization”, in Advances in Neural Information Processing Systems, Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K. Q., (ur.), Vol. 27. Curran Associates, Inc., 2014, dostupno na: https://proceedings.neurips.cc/paper/2014/file/feab05aa91085b7a8012516bc3533958-Paper.pdf

[32]Mu, J., Bhat, S., Viswanath, P., “Geometry of polysemy”, arXiv preprint arXiv:1610.07569, 2016.

[33]Schütze, H., “Automatic word sense discrimination”, Computational linguistics, Vol. 24, No. 1, 1998, str. 97–123.

[34]Reisinger, J., Mooney, R., “Multi-prototype vector-space models of word meaning”, in Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2010, str. 109–117.

[35]Rosseel, Y., “Mixture models of categorization”, Journal of Mathematical Psychology, Vol. 46, No. 2, 2002, str. 178–210.

[36]Love, B. C., Medin, D. L., Gureckis, T. M., “Sustain: a network model of category learning.”, Psychological review, Vol. 111, No. 2, 2004, str. 309.

[37]Griffiths, T., Canini, K., Sanborn, A., Navarro, D., “Unifying rational models of categorization via the hierarchical dirichlet process”, 2007.

[38]Huang, E. H., Socher, R., Manning, C. D., Ng, A. Y., “Improving word representations via global context and multiple word prototypes”, in Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2012, str. 873–882.

[39]Chen, X., Liu, Z., Sun, M., “A unified model for word sense representation and disambiguation”, in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, str. 1025–1035.

[40]Miller, G. A., “Wordnet: a lexical database for english”, Communications of the ACM, Vol. 38, No. 11, 1995, str. 39–41.

[41]Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., Weischedel, R., “Ontonotes: the 90% solution”, in Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers, 2006, str. 57–60.

[42]Neelakantan, A., Shankar, J., Passos, A., McCallum, A., “Efficient non-parametric estimation of multiple embeddings per word in vector space”, arXiv preprint arXiv:1504.06654, 2015.

[43]Liu, Y., Liu, Z., Chua, T.-S., Sun, M., “Topical word embeddings”, in Twenty-ninth AAAI conference on artificial intelligence, 2015.

[44]Yaghoobzadeh, Y., Kann, K., Hazen, T. J., Agirre, E., Schütze, H., “Probing for semantic classes: Diagnosing the meaning content of word embeddings”, arXiv preprint arXiv:1906.03608, 2019.

[45]Wijaya, D. T., Yeniterzi, R., “Understanding semantic change of words over centuries”, in Proceedings of the 2011 international workshop on DETecting and Exploiting Cultural diversiTy on the social web, 2011, str. 35–40.

[46]Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., Polosukhin, I., “Attention is all you need”, Advances in neural information processing systems, Vol. 30, 2017.

[47]Sennrich, R., Haddow, B., Birch, A., “Neural machine translation of rare words with subword units”, arXiv preprint arXiv:1508.07909, 2015.

[48]Ba, J. L., Kiros, J. R., Hinton, G. E., “Layer normalization”, arXiv preprint arXiv:1607.06450, 2016.

[49]Lekkas, A., Schneider-Kamp, P., Augenstein, I., “Multi-sense language modelling”, arXiv e-prints, 2020, str. arXiv–2012.

[50]Veli čković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., “Graph attention networks”, arXiv preprint arXiv:1710.10903, 2017.

[51]Miller, G. A., Leacock, C., Tengi, R., Bunker, R. T., “A semantic concordance”, in Human Language Technology: Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993, 1993.

[52]Ansell, A., Bravo-Marquez, F., Pfahringer, B., “Polylm: Learning about polysemy through language modeling”, arXiv preprint arXiv:2101.10448, 2021.

[53]Gururangan, S., Swayamdipta, S., Levy, O., Schwartz, R., Bowman, S. R., Smith, N. A., “Annotation artifacts in natural language inference data”, arXiv preprint arXiv:1803.02324, 2018.

[54]Bowman, S. R., Angeli, G., Potts, C., Manning, C. D., “A large annotated corpus for learning natural language inference”, arXiv preprint arXiv:1508.05326, 2015.

[55]Jacovi, A., Goldberg, Y., “Towards faithfully interpretable nlp systems: How should we define and evaluate faithfulness?”, arXiv preprint arXiv:2004.03685, 2020.

[56]Simonyan, K., Vedaldi, A., Zisserman, A., “Deep inside convolutional networks: Visualising image classification models and saliency maps”, arXiv preprint arXiv:1312.6034, 2013.

[57]Serrano, S., Smith, N. A., “Is attention interpretable?”, in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, str. 2931–2951.

[58]Wiegreffe, S., Pinter, Y., “Attention is not not explanation”, arXiv preprint arXiv:1908.04626, 2019.

[59]Pruthi, D., Gupta, M., Dhingra, B., Neubig, G., Lipton, Z. C., “Learning to deceive with attention-based explanations”, arXiv preprint arXiv:1909.07913, 2019.

[60]Rudin, C., “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”, Nature Machine Intelligence, Vol. 1, No. 5, 2019, str. 206–215.

[61]Mohankumar, A. K., Nema, P., Narasimhan, S., Khapra, M. M., Srinivasan, B. V., Ravindran, B., “Towards transparent and explainable attention models”, arXiv preprint arXiv:2004.14243, 2020.

[62]Tutek, M., Šnajder, J., “Staying true to your word:(how) can attention become explanation?”, arXiv preprint arXiv:2005.09379, 2020.

[63]Grimsley, C., Mayfield, E., R.S. Bursten, J., “Why attention is not explanation: Surgical intervention and causal reasoning about neural models”, in Proceedings of the 12th Language Resources and Evaluation Conference. Marseille, France: European Language Resources Association, May 2020, str. 1780–1790, dostupno na: https://aclanthology.org/2020.lrec-1.220

[64]Bibal, A., Cardon, R., Alfter, D., Wilkens, R., Wang, X., François, T., Watrin, P., CENTAL, I., “Is attention explanation? an introduction to the debate”, in Association for Computational Linguistics. Annual Meeting. Conference Proceedings, 2022.

[65]Neely, M., Schouten, S. F., Bleeker, M. J., Lucic, A., “Order in the court: Explainable ai methods prone to disagreement”, arXiv preprint arXiv:2105.03287, 2021.

[66]Denil, M., Demiraj, A., De Freitas, N., “Extraction of salient sentences from labelled documents”, arXiv preprint arXiv:1412.6815, 2014.

[67]Sundararajan, M., Taly, A., Yan, Q., “Axiomatic attribution for deep networks”, in International Conference on Machine Learning. PMLR, 2017, str. 3319–3328.

[68]Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., Samek, W., “On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation”, PloS one, Vol. 10, No. 7, 2015, str. e0130140.

[69]Shrikumar, A., Greenside, P., Kundaje, A., “Learning important features through propagating activation differences”, in International Conference on Machine Learning. PMLR, 2017, str. 3145–3153.

[70]Ribeiro, M. T., Singh, S., Guestrin, C., “" why should i trust you?" explaining the predictions of any classifier”, in Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, str. 1135–1144.

[71]Elman, J. L., “Finding structure in time”, Cognitive science, Vol. 14, No. 2, 1990, str. 179–211.

[72]Bengio, Y., Simard, P., Frasconi, P., “Learning long-term dependencies with gradient descent is difficult”, IEEE transactions on neural networks, Vol. 5, No. 2, 1994, str. 157–166.

[73]Pascanu, R., Mikolov, T., Bengio, Y., “On the difficulty of training recurrent neural networks”, in International conference on machine learning. PMLR, 2013, str. 1310–1318.

[74]Hochreiter, S., Schmidhuber, J., “Long short-term memory”, Neural computation, Vol. 9, No. 8, 1997, str. 1735–1780.

[75]Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., “Learning phrase representations using rnn encoder-decoder for statistical machine translation”, arXiv preprint arXiv:1406.1078, 2014.

[76]Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. *et al.*, “Language models are few-shot learners”, Advances in neural information processing systems, Vol. 33, 2020, str. 1877–1901.

[77]Vashishth, S., Upadhyay, S., Tomar, G. S., Faruqui, M., “Attention interpretability across NLP tasks”, arXiv preprint arXiv:1909.11218, 2019.

[78]Tutek, M., Snajder, J., “Toward practical usage of the attention mechanism as a tool for interpretability”, IEEE Access, 2022.

[79]Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., “Going deeper with convolutions”, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, str. 1–9.

[80]Srivastava, R. K., Greff, K., Schmidhuber, J., “Highway networks”, arXiv preprint arXiv:1505.00387, 2015.

[81]Hoerl, A. E., Kennard, R. W., “Ridge regression: Biased estimation for nonorthogonal problems”, Technometrics, Vol. 12, No. 1, 1970, str. 55–67.

[82]Tibshirani, R., “Regression shrinkage and selection via the lasso”, Journal of the Royal Statistical Society: Series B (Methodological), Vol. 58, No. 1, 1996, str. 267–288.

[83]Zou, H., Hastie, T., “Regularization and variable selection via the elastic net”, Journal of the royal statistical society: series B (statistical methodology), Vol. 67, No. 2, 2005, str. 301–320.

[84]Voita, E., Sennrich, R., Titov, I., “The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives”, in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, str. 4387–4397.

[85]Inan, H., Khosravi, K., Socher, R., “Tying word vectors and word classifiers: A loss framework for language modeling”, arXiv preprint arXiv:1611.01462, 2016.

[86]Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., Potts, C., “Recursive deep models for semantic compositionality over a sentiment treebank”, in Proceedings of the 2013 conference on empirical methods in natural language processing, 2013, str. 1631–1642.

[87]Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., Potts, C., “Learning word vectors for sentiment analysis”, in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Portland, Oregon, USA: Association for Computational Linguistics, Jun. 2011, str. 142–150, dostupno na: https://www.aclweb.org/anthology/P11-1015

[88]Johnson, A. E., Pollard, T. J., Shen, L., Li-wei, H. L., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., Mark, R. G., “Mimic-iii, a freely accessible critical care database”, Scientific data, Vol. 3, 2016, str. 160035.

[89]Brier, G. W., “Verification of forecasts expressed in terms of probability”, Monthly weather review, Vol. 78, No. 1, 1950, str. 1–3.

[90]Paninski, L., “Estimation of entropy and mutual information”, Neural computation, Vol. 15, No. 6, 2003, str. 1191–1253.

[91]Tishby, N., “The information bottleneck method”, in Proc. 37th Annual Allerton Conference on Communications, Control and Computing, 1999, 1999, str. 368–377.

[92]Sculley, D., “Web-scale k-means clustering”, in Proceedings of the 19th international conference on World wide web, 2010, str. 1177–1178.

[93]Hu, M., Liu, B., “Mining and summarizing customer reviews”, in Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, 2004, str. 168–177.

[94]Aharon, M., Elad, M., Bruckstein, A., “K-svd: An algorithm for designing overcomplete dictionaries for sparse representation”, IEEE Transactions on signal processing, Vol. 54, No. 11, 2006, str. 4311–4322.

[95]Williams, R. J., “Simple statistical gradient-following algorithms for connectionist reinforcement learning”, Machine learning, Vol. 8, No. 3, 1992, str. 229–256.

[96]Maddison, C. J., Mnih, A., Teh, Y. W., “The concrete distribution: A continuous relaxation of discrete random variables”, arXiv preprint arXiv:1611.00712, 2016.

[97]Zhang, B., Titov, I., Sennrich, R., “On sparsifying encoder outputs in sequence-to-sequence models”, in Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. Online: Association for Computational Linguistics, Aug. 2021, str. 2888–2900, dostupno na: https://aclanthology.org/2021.findings-acl.255

[98]Van Den Oord, A., Vinyals, O. *et al.*, “Neural discrete representation learning”, Advances in neural information processing systems, Vol. 30, 2017.

[99]Gumbel, E. J., Statistical theory of extreme values and some practical applications: a series of lectures. US Government Printing Office, 1954, Vol. 33.

[100]Maddison, C. J., Tarlow, D., Minka, T., “A* sampling”, Advances in Neural Information Processing Systems, Vol. 27, 2014.

[101]Bengio, Y., Léonard, N., Courville, A., “Estimating or propagating gradients through stochastic neurons for conditional computation”, arXiv preprint arXiv:1308.3432, 2013.

[102]Lei, T., Barzilay, R., Jaakkola, T., “Rationalizing neural predictions”, in Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Austin, Texas: Association for Computational Linguistics, Nov. 2016, str. 107–117, dostupno na: https://aclanthology.org/D16-1011

[103]Louizos, C., Welling, M., Kingma, D. P., “Learning sparse neural networks through l_0 regularization”, arXiv preprint arXiv:1712.01312, 2017.

[104]Kumaraswamy, P., “A generalized probability density function for double-bounded random processes”, Journal of hydrology, Vol. 46, No. 1-2, 1980, str. 79–88.

[105]Kingma, D. P., Welling, M., “Auto-encoding variational bayes”, arXiv preprint arXiv:1312.6114, 2013.

[106]Rezende, D. J., Mohamed, S., Wierstra, D., “Stochastic backpropagation and approximate inference in deep generative models”, in International conference on machine learning. PMLR, 2014, str. 1278–1286.

[107]Mnih, A., Gregor, K., “Neural variational inference and learning in belief networks”, in International Conference on Machine Learning. PMLR, 2014, str. 1791–1799.

[108]See, A., Liu, P. J., Manning, C. D., “Get to the point: Summarization with pointer-generator networks”, arXiv preprint arXiv:1704.04368, 2017.

[109]Williams, R. J., Zipser, D., “A learning algorithm for continually running fully recurrent neural networks”, Neural computation, Vol. 1, No. 2, 1989, str. 270–280.

[110]Graves, A., “Adaptive computation time for recurrent neural networks”, arXiv preprint arXiv:1603.08983, 2016.

[111]Sordoni, A., Bachman, P., Trischler, A., Bengio, Y., “Iterative alternating neural attention for machine reading”, arXiv preprint arXiv:1606.02245, 2016.

[112]Tutek, M., Šnajder, J., “Iterative recursive attention model for interpretable sequence classification”, arXiv preprint arXiv:1808.10503, 2018.

[113]Hashimoto, K., Xiong, C., Tsuruoka, Y., Socher, R., “A joint many-task model: Growing a neural network for multiple nlp tasks”, arXiv preprint arXiv:1611.01587, 2016.

[114]McCann, B., Bradbury, J., Xiong, C., Socher, R., “Learned in translation: Contextualized word vectors”, Advances in neural information processing systems, Vol. 30, 2017.

[115]Luong, T., Pham, H., Manning, C. D., “Effective approaches to attention-based neural machine translation”, in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, str. 1412–1421, dostupno na: https://aclanthology.org/D15-1166

[116]Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., Bengio, Y., “Maxout networks”, in International conference on machine learning. PMLR, 2013, str. 1319–1327.

[117]Munkhdalai, T., Yu, H., “Neural semantic encoders”, in Proceedings of the conference. Association for Computational Linguistics. Meeting, Vol. 1. NIH Public Access, 2017, str. 397.

[118]Dieng, A. B., Wang, C., Gao, J., Paisley, J., “TopicRNN: A recurrent neural network with long-range semantic dependency”, in ICLR 2016, 2016.

[119]Johnson, R., Zhang, T., “Supervised and semi-supervised text categorization using lstm for region embeddings”, in International Conference on Machine Learning, 2016, str. 526–534.

[120]Radford, A., Jozefowicz, R., Sutskever, I., “Learning to generate reviews and discovering sentiment”, arXiv preprint arXiv:1704.01444, 2017.

[121]Miyato, T., Dai, A. M., Goodfellow, I., “Adversarial training methods for semi-supervised text classification”, arXiv preprint arXiv:1605.07725, 2016.

[122]Kim, S., Lin, S., Jeon, S. R., Min, D., Sohn, K., “Recurrent transformer networks for semantic correspondence”, Advances in neural information processing systems, Vol. 31, 2018.

[123]Beltagy, I., Peters, M. E., Cohan, A., “Longformer: The long-document transformer”, arXiv preprint arXiv:2004.05150, 2020.

[124]Sinha, K., Jia, R., Hupkes, D., Pineau, J., Williams, A., Kiela, D., “Masked language modeling and the distributional hypothesis: Order word matters pre-training for little”, arXiv preprint arXiv:2104.06644, 2021.

[125]Box, G. E., Draper, N. R., Empirical model-building and response surfaces. John Wiley & Sons, 1987.

List of Figures

List of Tables

Biography

Martin Tutek was born on October 16, 1990 in Zagreb, Croatia. He received his B.Sc. in Computing from the University of Zagreb, Faculty of Electrical Engineering and Computing in 2012 and M.Sc. in Computer Science from the same university in 2014. From September 2014. until September 2015. he was employed as a computational linguist at the Joint Research Centre of the European Commission in Ispra. From February 2016, he is employed as a teaching and research assistant at the Department of Electronics, Microelectronics, Computer and Intelligent Systems at the Faculty of Electrical Engineering and Computing.

In September 2018, he was on a research visit to Data and Web Science Group of University of Mannheim, Germany. His research interests include natural language processing and machine learning, with a focus on model interpretability and explainability and learning multiprototype word representations. He is a member of the Association for Computational Linguistics.

List of publications

Journal publications

1. Tutek, M., Šnajder, J. "Toward Practical Usage of the Attention Mechanism as a Tool for Interpretability", IEEE Access, travanj 2022., str. 1-1.

Conference publications

- Tutek, M., Šnajder, J. "Staying True to Your Word:(How) Can Attention Become Explanation?", Proceedings of the 5th Workshop on Representation Learning for NLP, Srpanj 2020., str. 131–142.
- Tutek M., Šnajder J. "Iterative recursive attention model for interpretable sequence classification", Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, Studeni 2018., str. 249–257.
- Di Buono MP, Šnajder J, Baši ć BD, Glavaš G, Tutek M, Milić-Frayling N. "Predicting news values from headline text and emotions", Proceedings of the 2017 EMNLP Work-

shop: Natural Language Processing meets Journalism, Rujan 2017., str. 1–6.

•Rotim L, Tutek M, Šnajder J. “Takelab at semeval-2017 task 5: Linear aggregation of word embeddings for fine-grained sentiment analysis of financial news”, Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), Kolovoz 2017., str. 866–871.

•Di Buono MP, Tutek M, Šnajder J, Glavaš G, Baši ć BD, Milić-Frayling N. “Two layers of annotation for representing event mentions in news stories”, Proceedings of the 11th Linguistic Annotation Workshop, Travanj 2017., str. 82–90.

•Tutek M, Glavaš G, Šnajder J, Mili ć-Frayling N, Dalbelo Bašić B. “Detecting and ranking conceptual links between texts using a knowledge base”, Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, Listopad 2016., str. 2077–2080.

•Tutek, M., Sekuli ć, I., Gombar, P., Paljak, I., Čulinović, F., Boltužić, F., Karan, M., Alagić, D. and Šnajder, J., Takelab at semeval-2016 task 6: “Stance classification in tweets using a genetic algorithm based ensemble”, Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016), Lipanj 2016., str. 464-468.

Životopis

Martin Tutek rođen je 16. listopada 1990. u Zagrebu, Hrvatska. Preddiplomski studij računarstva završio je u 2012. godine na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu, na kojem je i 2014. godine završio diplomski studij računarske znanosti. Od rujna 2014. godine do rujna 2015. godine zaposlen je kao računalni lingvist u istraživačkom centru Europske komisije u Ispri. Potom, od veljače 2016. zaposlen je zaposlen kao asistent i znanstveni suradnik na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu.

U rujnu 2018. godine boravio je u inozemstvu na znanstvenom usavršavanju u Grupi za podatkovnu i internetsku znanost Sveučilišta u Mannheimu, Njemačka. Njegovi istraživački interesi uključuju obradu prirodnog jezika i strojno učenje, s fokusom na interpretabilnost i objašnjivost modela baziranih na neuronskim mrežama te učenje višeprototipnih reprezentacija riječi. Član je strukovne udruge ACL (Association for Computational Linguistics).