Bachelor's thesis

Bachelor's Programme in Computer Science

The Transformer Model and Its Impact on the Field of Natural Language Processing

Matias Nieminen

June 28, 2023

Faculty of Science

University of Helsinki

Contact information

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki,Finland

Email address: info@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Study programme
Faculty of Science	Bachelor's Programme in Computer Science

Tekijä — Författare — Author

Matias Nieminen

Työn nimi — Arbetets titel — Title

The Transformer Model and Its Impact on the Field of Natural Language Processing

Ohjaajat — Handledare — Supervisors

Dr. Andrew Rebeiro-Hargrave

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Bachelor's thesis	June 28, 2023	34 pages

Tiivistelmä — Referat — Abstract

Natural Language Processing is a hot topic in today's Artificial Intelligence community for its rapid development in the past decade. It covers areas of computation that aim to derive insight from natural languages such as English or Sami in order to utilize them in subsequent processes. Due to the developments, the field is increasingly important and is expected to impact the world around us dramatically in the coming years. This thesis reviews the history of language processing with a selection of some of the most influential models like recurrent and convolutional neural networks. The Transformer model is unraveled and its performance is reviewed in a set of varying tasks. Finally, recent developments like pre-trained transformers, multi-modality, and language generation are inspected.

ACM Computing Classification System (CCS)
Computing methodologies → Artificial intelligence → Natural language processing
Discourse, dialogue and pragmatics

Avainsanat — Nyckelord — Keywords

natural language processing, Transformer, neural network, artificial intelligence, convolution, recurrence

Säilytyspaikka — Förvaringsställe — Where deposited

Helsinki University Library

Muita tietoja — övriga uppgifter — Additional information

Software study track

Lyhennelmä

Vuonna 2017 julkaistun Attention Is All You Need -nimisen tutkimuksen jälkeen luonnollisen kielen prosessointi (NLP) on muuttunut lähes täysin. Suurin osa tästä kehityksestä saa kiittää Transformer-arkkitehtuuria, mikä on edellämainitun tutkimuksen esittelemä NLP-menetelmä. Alunperin kyseistä arkkitehtuuria esiteltiin käytettäväksi kielten kääntämiseen ja tulkkaamiseen, mutta sittemmin sitä on käytetty käytännössä kaikilla NLP:hen liittyvillä alueilla. Tämä tutkielma toimii lyhyenä johdantona luonnollisen kielen käsittelyn maailmaan, sen viimeaikaiseen kehitykseen ja siihen, miten Transformer-pohjaiset mallit ja tekniikka nimeltä huomio (attention) ovat muuttaneet ko. tutkimusalueen jopa laajalti kaupallisesti kannattavaksi alaksi, jollainen se nykypäivänä on. Nämä toimivat myös tutkielman tutkimuskysymyksinä.

Luonnollisen kielen käsittely on tutkimuksen ja sovellusten alue, jossa tutkitaan, miten tietokoneita voidaan käyttää ymmärtämään ja käsittelemään luonnollisen kielen tekstiä tai puhetta, sekä tekemään tällä ymmärryksellä hyödyllisiä asioita. Luonnollisen kielen käsittelyä tehdään lukemattomilla eri tavoilla, joista yksi on kontekstivapaa kielioppi (Context-Free Grammar), jota käytetään useimmiten kääntäjissä ohjelmointikielten jäsentelyssä. Muita NLP:n käyttökohteita ovat konekääntäminen (Machine Translation), chatbotit ja puheen tunnistaminen (Speech Recognition). Kehittyneelle NLP:lle on lähes rajattomasti sovelluksia, jotka vaihtelevat lauseiden luomisesta tiivistelmiin ja vuorovaikutukseen kaikenlaisten tietokonejärjestelmien kanssa. Yhteenvetona NLP on tutkimusalue, joka keskittyy mallintamaan sitä, miten ihmiset käyttävät ja ymmärtävät luonnollista kieltä, jotta sitä voidaan hyödyntää erilaisten tehtävien laskennassa.

Yksi tärkeimmistä osista tutkielmassa on lukijan johdattaminen NLP:hen käsittelemällä ensin sen historiaa. NLP:n historiasta nostetaan joitakin kaikista merkittävimpiä tapahtumia ja malleja, joilla NLP:tä tehtiin ennen Transformer-mallin esittelyä. NLP:tä on tehty jo 1940-luvulta asti, milloin sen kehitys oli todella hidasta alhaisten resurssien ja tehottomien tietokoneiden takia. 1960-luvulla suurin osa rahoituksesta katkaistiin, sillä tuloksia joita oli toivottu ei yksinkertaisesti saavutettu. Siitä huolimatta ennen 1980-lukuun mennessä oli kehitetty esimerkiksi SHRDLU-niminen simuloitu robottikäsi, joka pystyi vuorovaikuttamaan kolmiulotteiseen palikkamaailmaan käyttäjän antamilla luonnollisen kielen käskyillä. Samoihin aikoihin julkaistiin järjestelmä nimeltä LUNAR, jonka avulla käyttäjä saattoi olla vuorovaikutuksessa tietokannan kanssa. Kummatkin näistä projekteista olivat kehittyneitä aikanaan, mutta toimivat chatbotit ja konekääntäjät olivat vielä pitkän työn takana.

Keinotekoisia neuroverkkoja taas ei juurikaan käytetty ennen 2010-lukua siitä huolimatta, että niiden käyttäminen alkoi olla laskennallisesti realistista. Sen sijaan suosituimpia teknologioita ja malleja NLP:hen olivat lineaariset mallitu kuten logistinen regressio ja tukivektorikoneet (Support Vector Machines). Nämä mallit ovat binäärisiä luokittelijoita, jotka käyttävät pääasiassa hyvin moniulotteista dataa ennusteiden tekemiseen. Mallien lineaarisuus myös tarkoittaa, että jotta mallit pystyvät mallintamaan jotakin tietoa tarkasti, pitää tiedon sisältävien pisteiden olla jaettavissa suoralla viivalla tai tasolla niiden esitysavaruudessa. Teoriassa kaikki tieto voidaan tällaiseen formaattiin muuntaa, mutta se usein vaatii jo valmiiksi moniulotteisen tiedon muuttamista yhä moniulotteisemmaksi. Tämä taas vaatii lisää laskentatehoa, ja ei siksi ole käytännön kannalta robusti ratkaisu.

2010-luku oli syväoppimisen (Deep Learning) ja keinotekoisten neuroverkkojen (Artificial Neural Networks) vuosikymmen, ja sama trendi on jatkunut myös 2020-luvulle. Scopus, joka on Elsevierin viittaustietokanta, sisältää 3500 tutkimusta jotka mainitsevat syväoppimisen 2000-luvulta, mutta 135000 samat kriteerit täyttävää tutkimusta 2010-luvulta. Tämä havainnollistaa hyvin syväoppimisen popularisaation ko. aikakautena. 2010-luvulla alettiin hyödyntää monia jo aiemmin keksittyjä malleja kuten konvoluutioverkot (Convolutional Neural Networks) ja pitkäkestoinen lyhytkestomuisti -arkkitehtuuri (Long Short-Term Memory networks), joilla päästiin siihenastisiin huipputuloksiin eri tehtävissä, kuten tunneanalyysissä ja konekääntämisessä. Lisäksi "moniuloitteisuuden kirouksesta" päästiin eroon niin kutsutuilla sanojen upotuksilla (word embeddings).

Seuraavaksi tutkielmassa esitellään ennen Transformer-arkkitehtuurin esittelemistä käytetyt huipputason suorituskykyä demonstroivat mallit kuten konvoluutioneuroverkot, takaisinkytkeytyvät neuroverkot (Recursive Neural Networks, RNNs) ja pitkäkestoiset lyhytkestomuistiarkkitehtuurit. Tätä ennen kuitenkin tutustutaan yksinkertaisimpaan neuroverkon muotoon, Perceptroniin. Perceptron esiteltiin 1958-vuonna Frank Rosenblattin toimesta Cornellin ilmailualan laboratoriossa. Sen inspiraationa toimi biologinen silmä ja siinä oli vain neljä yksinkertaista komponenttia; Syötteet, painot, vinouma, ja itse neuroni. Neuronissa taas on aktivaatiofunktio, joka päättää neuronin aktivoitumisesta, eli toisin sanoen siitä, mitä Perceptroni ajattelee. Syötteet koostuvat datasta, mitä neuroverkolle annetaan. Syötteet ovat numeerisessa muodossa, ja kaikilla niillä on vastaava paino jolla syöte kerrotaan ennen sen syöttämistä neuronille. Kun skaalatut syötteet saapuvat neuroniin, niiden yhteinen summa lasketaan ja siihen lisätään vielä vinouma. Vinouma on staattinen luku, joka antaa neuroverkoille lisää ilmaisukykyä. Lopuksi neuroni antaa summan aktivaatiofunktiolle, joka jolle yleinen operaatio on sigmoid-funktio. Sigmoid-funktion ulostulo on

Perceptronin tapauksessa myös sen itsensä lopullinen ulostulo. Jotta Perceptroni ja sitä suuremmatkin neuroverkot saataisiin antamaan oikeita vastauksia, pitää aiemmin mainintut painot viilata oikeiksi. Tämä tapahtuu koulutusdatan ja ns. vastavirta-algoritmin (Backpropagation algrithm) avulla, jossa ensin annetaan Perceptronille jotkin satunnaiset painot, ja sitten sen antaman oikean tai väärän vastauksen avulla vastavirta-algoritmi muuttaa painoja oikean vastauksen kannalta suotuisimmiksi.

Takaisinkytkeytyvät neuroverkot ovat neuroverkkoja, jotka toteuttavat rekursion solmujen välillä, mahdollistaen palautetun yhteyden ja kontekstin sisällyttämisen soluihin. Ne soveltuvat hyvin sekvenssimuotoisen ja ajallisesti muuttuvan datan mallintamiseen, ja niillä on sovelluksia monilla eri aloilla. Toisin kuin eteenpäin syötetyt neuroverkot (Feedforward Neural Networks) kuten Perceptroni, takaisinkytkeytyviä neuroverkkoja käytettäessä ei tarvitse valita tiettyä kontekstikokoa, mutta täydellistä ääretöntä kontekstia ei voi käyttää gradienttiin liittyvien haasteiden vuoksi. Gradientilla tarkoitetaan painon päivittävää arvoa, joka lähtee käsistä takaisinkytkentöjen takia, johtaen siihen, ettei neuroverkko opi. Kontekstilla tarkoitetaan tässä sitä, montako sanaa tai "tokenia" neuroverkko muistaa. RNN:illä on rajoitteita, kuten pitkät koulutusajat, monimutkaiset vastavirta-algoritmit ja häviävät/räjähtävät gradientit, mikä tekee niistä vähemmän optimaalisia pitkän aikavälin riippuvuuksien oppimiseen. Erilaisia RNN-arkkitehtuureja ovat muun muassa monesta moneen, monesta yhteen, yhdestä moneen ja yhdestä yhteen, joissa määrät viittaavat syötteiden ja ulostulojen suhteeseen. Takaisinkytkeytyvän neuroverkon purkaminen vastaavaksi äärelliseksi, eteenpäin syötetyksi neuroverkoksi mahdollistaa koulutuksen ajan läpi kulkevalla vastavirta-algoritmilla (Backpropagation Through Time), mutta gradienttiin liittyvät ongelmat jatkuvat.

Gradienttien ongelma on ratkaistu ns. pitkäkestoinen lyhytkestomuisti -arkkitehtuureilla, jotka käyttävät portteja (gates) määrittelemään, säilytetäänkö jokin saatu syöte solutilassa (cell state) vai ei. Tämä estää räjähtävät ja katoavat gradientit. Tästä huolimatta arkkitehtuurin konteksti-ikkuna on rajattu noin kahteensataan tokeniin, sillä pitkän välin riippuvuuksissa pitää luottaa arkkitehtuurin välimuistiin, joka ei kykene erottelemaan esimerkiksi sanajärjestystä viidenkymmenen sanan ulkopuolella. Lisäksi rinnakkaislaskentaa ei pystytä tehokkaasti hyödyntämään käytettäessä mallia, sillä jokainen ulostulo riippuu edellisestä ulostulosta sekä siihen liittyvästä neuroverkon tilasta.

Konvoluutioneuroverkot ovat myös eteenpäin syötettyjä neuroverkkoja, joiden tärkein ominaisuus on niiden konvoluutiokerrokset. Konvoluutiokerrokset tunnistavat kuvioita syötteessä. Tämän takia ne soveltuvat ensisijaisesti käytettäväksi konenäköön, mutta tästä

huolimatta ne tuottavat hyviä tuloksia myös kielen prosessoinnissa. Toisin kuin takaisinkytkevät neuroverot, konvoluutioverkkojen kanssa voidaan helposti käyttää rinnakkaislaskentaa. Rajoitteita konvoluutioneuroverkoilla kuitenkin on pitkän matkan riippuvuuksien kanssa, jonka lisäksi ne tarvitsevat poikkeuksellisen suuren määrän dataa oppimiseen.

Ennen Transformer-mallin esittelyä huipputulokset tulivat joko konvoluutioverkkojen tai takaisinkytkevien verkkojen toimesta. Tulokset kuitenkin kärsivät, sillä molemmat mallit kärsivät huomattavista ongelmista joko konteksti-ikkunan tai rinnakkaislaskennan kanssa. Konvoluutioverkot eivät muista kontekstia, takaisinkytkevät taas muistavat sen hieman paremmin, mutta niiden laskenta-ajat nousevat suuren tietomäärän puitteissa niin korkeiksi, että ne eivät ole käytännöllisiä. Transformer-malli vastaa kumpaankin näistä ongelmista elegantisti. Alkuperäinen Transformer-malli on eteenpäin syötetty neuroverkko jonka kanssa voidaan hyödyntää rinnakkaislaskentaa tehokkaasti, ja joka sen lisäksi muistaa hätkähdyttävän pitkän konteksti-ikkunan verrattuna edellisiin ratkaisuihin vakiolla aikakompleksisuudella riippuvuuden etäisyyteen verrattuna. Syötteen koko kuitenkin vaikuttaa sen aikakompleksisuuteen neliöllisesti.

Transformer-malli on siis kokonaisuus, joka koostuu monesta eteenpäin syöttävästä neuroverkosta ja tekniikasta tunnistaa sanojen välisiä riippuvuuksia tekniikalla jota kutsutaan "huomioksi". Sen arkkitehtuuri perustuu enkooderi-dekooderi-arkkitehtuuriin, missä ensin enkooderi luo numeraalisen version sille annetuista syötteistä. Nämä numeraaliset esitykset, eli sanaupotukset, sisältävät tiedon sanan semanttisesta merkityksestä sekä sanan suhteellisesta sijainnista syötteessä. Tutkielmassa käydään läpi suhteellisen yksityiskohtaisesti miten tieto kulkee Transformer arkkitehtuurin läpi. Enkooderi muodostuu kahdesta päätasosta; Ensimmäisenä vuorossa on niin sanottu "Multi-head attention", joka tarkoittaa sanaupotuksen rinnakkaista prosessointia huomiomekanismilla. Jokainen huomiomekanismin instanssi tekee johtopäätöksensä kolmen eri vektorin avulla, jotka lasketaan sanaupotuksesta käyttäen eteenpäin syötettyjä neuroverkkoja. Jokaisen instanssin ulostulo summataan sitten yhteen keskenään, jonka lisäksi alkuperäinen sanaupotus summataan tästä saatuun vektoriin. Toinen taso saa syötteenään tämän summaoperaation normalisoidun tuloksen. Se koostuu vain yhdestä eteenpäin syötetystä neuroverkosta. Myös toisen tason ulostulo summataan tason syötteen kanssa — eli edellisen tason ulostulon kanssa — ennen lopullisen ulostulon syöttämistä dekooderille. Tällaisia kahden tason enkooderitasoja taas on 6 peräkkäin alkuperäisessä Transfomer-mallissa.

Seuraavaksi enkooderitasojen ulostulo annetaan dekooderille. Dekooderin tehtävä on muuttaa sen syöte takaisin tekstiksi. Dekooderi koostuu samantapaisista osista, mutta sille

annetaan lisäksi koko Transformerin aiemmin tuottamat ulostulot. Tämä tekee Transformerista autoregressiivisen mallin, eli sille syötetään takaisen sen itse tuottamat ulostulot, kunnes se viimein antaa ulos "end"-tokenin. Toisin sanoen, Transformer raksuttaa ikuisesti, kunnes se päättää olevansa valmis.

NLP:ssä käytetään suorituskyvyn mittaamiseen yleisimmin tarkkuutta (precision), herkkyyttä (recall), ja näistä kahdesta vedettyä F1-pisteytystä. Tämän lisäksi konekääntämisessä käytetään BLEU-pisteytystä, mikä määritellään konekäännöksen samankaltaisuudella ihmisen kääntämään ekvivalenttiin. BLEU:ssa arvion ollessa 1 on ihmisen käännös identtinen konekäännökseen. BLEU ilmoitetaan useimmiten prosenttina, eli skaalalla 0-100.

Transformerin vaikutus NLP:hen on ollut hyvin näkyvä. Alkuperäinen Transformer-malli löi sen aikaiset ennätykset WMT2014 tietoaineiston English-to-German konekääntämistehtävissä selkeällä 2.0 BLEUn erolla edelliseen ennätykseen. Tähän Transformer-malli käytti vain pienen osan koulutuksenaikaisesta laskennallisesta tehosta. English-to-French tietoaineistolla se saavutti 0.44 BLEU:n parannuksen käyttämällä vain neljänneksen koulutusajasta.

Alkuperäiset tulokset eivät saa kuitenkaan hämätä, sillä Transformerin esittelystä lähtien samaan arkkitehtuuriin perustuvat mallit kuten Bidirectional Encoder Representations from Transformers (BERT) ja Generative Pre-trained Transformer (GPT) ovat dominoineet alaa. Näiden lisäksi on lukemattomia muita avoimen lähdekoodin malleja kuten Transformer-XL, jotka dominoivat ennätyksiä suorituskykymittauksissa. Tutkielmassa tutkitaan 3:ssa eri tehtävässä parhaiten suoriutuvia malleja. Nämä tehtävä ovat: Tekstin luokittelu, Kysymys-Vastaus-parit, ja konekääntäminen. Nämä tehtävät valittiin, sillä ne edustavat Transformer-mallin monikykyisyyttä.

Tekstin luokittelussa tutkielma katsoo IMDb-mittausta, missä IMDb-arvostelutietovarannosta pyritään tunnistamaan arvostelun pääasiallinen tunne. Tämänkaltaisissa analyyseissä tukivektorikoneet ovat tuottaneet tyypillisesti parhaita tuloksia, mutta sittemmin Transformer-pohjainen malli nimeltä Transformer-XL on ottanut ykkössijan loistavalla 97.1:n tarkkuudella. Kysymysten vastauksessa Transformer on luonnollinen valinta, ja siihen perustuva malli XLNet saavuttaa 95.08:n F1-pisteytksen. Tämän lisäksi ANNA-niminen Transformer-pohjainen malli on saavuttanut jopa 95.719: F1-pisteytyksen, mutta kyseistä julkaisua ei ole vertaisarvioitu. Konekääntämiseen Transformer on jälleen luonnollinen valinta sen alkuperäisen tarkoituksen takia, ja siinä kaikki parhaat 5 mallia ovat Transformer-pohjaisia malleja. Parhaat tulokset saa erityinen Transformerille kehitetty tekniikka nimeltä parametrien jako. Kyseinen tekniikka saavuttaa WMT2014:n English-to-German tieto-

varannolla 35.14 BLEU pisteet. Tämä on siis huomattava kehitys alkuperäiseen Transfomer-malliin, joka sai samasta tehtävästä pisteet 28.4 BLEU.

Sen lisäksi, että Transformer-pohjaiset mallit dominoivat suoritukykymittauksia, niiden tehokkuus on tehostanut aiemmin hankalia konsepteja. Näihin kuuluu esimerkiksi ennalta koulutetut mallit, joita kehittäjät voivat sitten hienosäätää omiin tarkoituksiinsa sopiviksi. Näistä malleista tunnetuin esimerkki lienee GPT-perhe, joista GPT-2-mallin lähdekoodi on avoimesti saatavilla. Toinen esimerkki on jo aiemmin keskusteltu BERT. Tällaisten mallien vapaa saatavuus on muuttanut osaltaan NLP:tä kaupallisessa maisemassa. Transformer-mallin tehokkuus avasi myös reittejä generatiivisessa tekoälyssä, mikä näkyy esimerkiksi Transformer-mallien ylivertaisuudessa kysymys-vastaus-tietovarantojen suorituskykymittauksissa. Suurimmillaan Transformer-mallit saattavat olla jopa satojen miljardien parametrien malleja. Tällaisesta esimerkki on GPT-3.5, missä on 175 miljardia koulutettavaa parametria. Näin Transformer-mallit pystyvät toimimaan myös useilla eri kielillä, sekä ottamaan tietoa vastaan samanaikaisesti kuvaformaatissa. Tästä, ja edellämainittujen Transformerien kilpailuetujen ansiosta kielimallien generatiivinen ilmaisuvoima on noussut huomattavasti. Tämä on johtanut kaupalliseen läpimurtoon.

Transformer-malli on siis johtanut NLP-alan murrokseen tutkituilla osa-alueilla. Tähän syyt ovat olleet Transformer-mallin loistava huomiomekanismi, rinnakkaistettavuus, pitkän matkan riippuvuuksien käsittely, siihen perustuvien mallien laaja adoptio tutkimuksessa ja teollisuudessa, sekä syväoppimismetodien popularisaatio.

Contents

1 Introduction

After the famous paper Attention Is All You Need [56] was published in 2017, Natural Language Processing (NLP) has come a long way. Much of this development can be attributed to the Transformer model, which was the paper's novel proposition in Neural Network (NN) model architecture. The Transformer's use cases have expanded to almost all areas of Artificial Intelligence, despite initially being proposed for machine translation tasks [56]. While the Transformer has proven useful in a broader range of problems, this thesis focuses its attention on the notorious model's impact on the field of NLP. This thesis will serve as a brief introduction to the world of NLP, its recent developments, and how Transformer-based models and a technique called attention have transformed the area of research into the commercially viable trend that it is today.

A concise definition for NLP is provided in [10]: "Natural Language Processing is an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to do useful things". NLP was initially utilized at an intersection of artificial intelligence and linguistics, although, not all NLP is in the context of artificial intelligence. NLP is conducted in a myriad of ways, one of which is Context-Free Grammar, which is most often used in compilers in parsing programming languages [44]. Other uses of NLP include machine translation, chatbots, and speech-to-text. There are near-infinite applications for advanced NLP, ranging from generating sentences to summarizing papers to interacting with computer systems of all kinds. In summary, NLP is a research area concentrated on modeling how humans use and understand natural language in order to utilize it in the computation of various tasks.

One of the most prevalent applications of NLP is Large Language Models (LLMs). LLMs are programs that attempt to model and process language in various ways often utilizing Deep Learning (DL) techniques, some of which are described in chapter 2. LLMs are used in all areas of NLP and can be trained on vast amounts of sequential data. Before the Transformer network architecture was introduced, state-of-the-art LLMs were composed of a combination of convolutional and recurrent network layers. Similarly to the Transformer, they also made use of encoder-decoder architectures. Due to advances in LLMs in recent years, the area of NLP has taken tremendous leaps. LLMs such as OpenAI's Generative Pre-trained Transformer (GPT) and Google's Bidirectional Encoder Representations from

Transformers (BERT) have taken the world by storm by breaking the barrier between everyday consumer utility and Artificial Intelligence (AI).

At the heart of the growth in LLMs is the Transformer model, introduced in late 2017 by researchers at Google Brain, Google's deep learning research unit. The Transformer model uses a technique called self-attention that allows the model to draw distinctions between tokens in sequential data with more specificity [3]. While the Transformer is not the first model to use an attention mechanism, it was the first to rid itself of using so-called convolutional and recurrent layers. Despite the dispensed complexity, the Transformer quickly performed better in a number of tests used to approximate the capability of a language model. Among other benchmarks, these tests included WMT 2014 English-To-German and English-To-French translation tasks that are designed to evaluate the state of the art in machine translation tasks [6].

Generalizability was evaluated with tests on English constituency parsing, in which the model analyzes sentences by breaking them down into parse trees of Part-Of-Speech (POS) tags. The Transformer model was able to perform better than all previously reported models with an exception of Recurrent Neural Network Grammars [56], a model that explicitly models nested, hierarchical relationships between words and phrases [18].

Transformer-based models are not the only contributor to the rise of impressive LLMs. Other factors like the availability of data through the internet, continuously increasing computational power, sophisticated optimization algorithms, and more all play a part in making LLMs more robust. To unravel the impact of the Transformer model, we will briefly look into the history of language modeling, how the Transformer model is composed and how it processes data on a grassroots level, how state-of-the-art language modeling has been affected by the integration of Transformer-based models, and what effects this has had on the field in the context of research, and commerce. We will take a look at the most influential models and the architectures behind them, how they have affected the landscape, and how they have been utilized.

2 Historical Overview

NLP has a long history of different kinds of solutions to different kinds of problems. To understand the impact of the Transformer, this section covers briefly a selection of innovations and solutions that have been presented before the popularization of DL and the Transformer.

2.1 Toward Deep Learning

Natural Language Processing dates back to the 1940s. Many of the challenges that researchers faced in the dawn of NLP have been wiped out simply through technological advances like the dramatic increase of computational power. Despite its long history, the focus has been geared toward machine translation. Not much progress was made in the first two decades of NLP, which lead to much of the funding for machine translation projects being cut in 1966. Nevertheless, notable systems developed after 1966 and 1980 include SHRDLU, which is a simulation of a robot arm that could respond to a limited amount of commands given in natural language that regard a simple world of blocks that the robot arm could interact with. Users could ask questions, make statements, and give orders to SHRDLU. In terms of the generality of a model, SHRDLU was a significant feat at its time in 1972 [17]. Around the same time, a database interface system titled LUNAR was demonstrated for the first time. LUNAR was based on Augmented Transition Networks and Woods' Procedural Semantics. Approaching the 80s, the most notable advances in the field of NLP were related to the syntax of different languages. Language models were still based on complex rule sets and much less progress had been made than what was anticipated in the 1960s [29]. While the first Artificial Neural Network (ANN) was introduced early on in the 1940s [1], significant results in NLP via ANNs only began to appear in 2000, when [5] proposed a feed-forward neural network to be used in conjunction with a lookup table. In the conducted tests, the ANN performed substantially better than the standard trigram language model that was used at the time. Truly conversational language models were still beyond the horizon, but more papers concerning deep learning methods in NLP began to appear.

Despite advancements in computation and ANNs in the first decade of the 21st century,

NLP was dominated by linear models such as Support Vector Machines (SVMs) and logistic regression that used high dimensional data and sparse vectors to represent tokens. Despite being popular in NLP, SVMs are binary classifiers, which means that they can output one of two values. This requires the data to be in a linearly separable format, which means that the represented classes of data can be divided on opposite sides of a straight line or plane. Moreover, SVMs can transform non-linear data to be linearly separable using kernel functions. A kernel function is simply an operation that is performed on the data points in order to transform the data into a desired shape. Any data can be transformed to be linearly separable, but transforming it requires the SVM to increment the dimensionality of the data. Increasing the dimensionality of the data will lead to the number of possible solutions increasing exponentially [23], which limits the SVM significantly in more complex relationships between variables. Similarly, other linear models are not sufficient in modeling real-world data, as strictly linear dependencies are rarely present in real-world scenarios. Some of the more influential papers before the 2010s were [47] and [12], which serve as good estimations of the state of NLP in the early 2000s.

In [47] Pang et al. examine whether sentiment classification can be treated as a special case of the already established genre of topic classification. Pang et al. employ the common classification methods of the 2000s, Naive Bayes, Maximum Entropy, and SVM classifiers. Their approach, while outperforming the human-produced baselines, can't perform to the standard of topic classification. Pang et al. argue that more subtle analysis and more sophisticated algorithms are required so that the focus of a sentence can better be determined [47]. In 2008, as computation was allowing larger NNs to be built, [12] introduced a Convolutional Neural Network (CNN) that, given a sentence, would output different predictions such as POS tags, chunks (syntactic constituent labeling). The architecture defined by [12] wanted to overcome the typical failings of shallow, linear classifiers. The CNN was able to improve upon previous NNs and parse tree-based solutions in semantic role labeling, which Collobert et al. defined as the most important task of their model. Further advancements lead to the emergence of NNs as viable tools for all kinds of machine-learning tasks. Some of the earlier advancements sprung from computer vision applications, but have spread onto NLP applications with increasingly complex models being proposed for complex NLP problems [60].

2.2 Re-emergence of Deep Learning

Deep learning has become an increasingly popular research topic. We queried Scopus, Elsevier's citation database, and found 135,000 papers that mention it appearing from 2011 to 2020. A stark contrast to the 3,500 papers mentioning deep learning found from 2001 to 2010 within the same database. State-of-the-art solutions began to emerge in computer vision and NLP with models such as CNNs, Long Short-Term Memory networks (LSTMs), and Recurrent Neural Networks. One of the major contributions to the rapid success of DL in NLP is the reduction of the dimensions of word representations. The reduction was achieved with distributed word representations called word embeddings. The most influential paper on such representations was [43], which introduced extensions to a word embedding model called skip-gram, making it faster and producing more regular word representations. Skip-gram and CBOW were initially proposed by [42] in their 2013 paper and are themselves simple fully connected neural networks. The following is a brief introduction to the most common NN architectures before the Transformer was introduced.

2.3 The Perceptron

Neural networks consist of units called neurons that are connected to each other by weights. In order to better understand the architectures of larger and more complex NNs, let's look at the most simple NN there is, The Perceptron, which was introduced in 1958 by Frank Rosenblatt at Cornell Aeronautical Laboratory [50]. The Perceptron can be thought of as a Feedforward Neural Network (FNN) which is limited in its ability and can only produce linear classifiers, due to not utilizing the added complexity of hidden layers. However, The Perceptron serves as an excellent example of the most basic components in NNs.

2.3.1 Architecture of The Perceptron

The Perceptron consists of a single neuron and is composed of 4 main parts; The inputs $\vec{x}$, weights $\vec{w}$ and a bias b, the neuron that is connected to the inputs by the weights, and the activation function. The inputs can be represented as a one-dimensional vector of real values, where each of the values represents a feature. A single input does not provide feedback to neurons preceding it, as there is only one neuron, which is why The Perceptron is an FNN. There is no limit on how many features can be fed into the neuron. After the

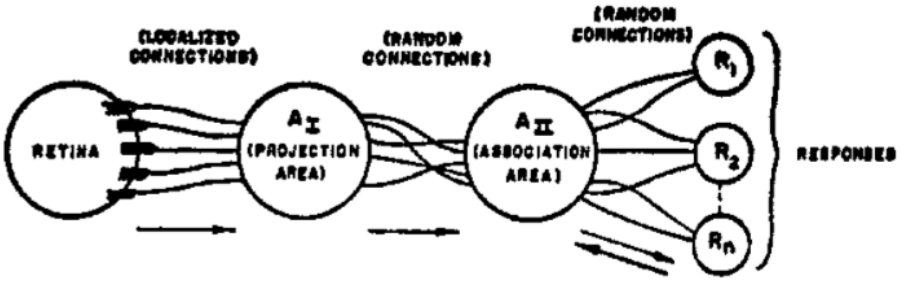

Figure 2.1: Organization of the Perceptron, [50]

.

inputs have traversed to the neuron through the synaptic weights, the neuron performs a weighted sum operation. The weighted sum can be defined as the following:

$$weighted\,sum = \sum_{i=1}^{k} (w_i x_i)$$

where k denotes the number of features and w_i denotes the weight corresponding to the input x_i.

After the weighted sum is calculated in the neuron, the bias b is added to the total. The bias enables the Perceptron to reach a better result by allowing it to approximate any linear function in the classification. The result of the weighted sum is forwarded to the activation function. In our example, the activation function is defined as the Sigmoid function:

$$activation(input) = \frac{1}{1 + e^{-input}}$$

but other functions can be used. The activation function produces the final output of the Perceptron, which is why the Sigmoid function is a natural choice, as a single Perceptron is only capable of binary classification. The Perceptron, much like any other neural network, can learn by adjusting its weights based on training data and a method called backpropagation.

2.3.2 Backpropagation

Backpropagation is an algorithm that tunes the weights of an NN based on the distance between the NN's prediction and its target. An NN can be trained by running backpropagation on each of the examples in the training set, thus iteratively approaching the

optimum values for the weights. Modern backpropagation is derived from Automatic Differentiation [4], which was introduced by a Finnish university student Seppo Linnainmaa in 1970 in his Master's thesis [38]. In ML and DL, the loss function is performed on the final output of the model to determine the model's performance on said output. Training data can consist of a series of tuples (x, y), where x denotes a data point from which a prediction is to be derived, with the correct prediction being y. The loss function of a NN takes the final output, or prediction, of the network and determines the *cost* associated with the prediction by measuring the distance between the prediction and the target y. A commonly mentioned loss function for classification tasks is cross-entropy. The backpropagation algorithm being utilized in the training uses the output of the cost function to determine the gradient of each of the layers' weighted input in order to adjust each of the weights so that the output of the cost function is minimized.

2.4 Recurrent Neural Networks

Recurrent Neural Networks are NNs that implement recursion between the nodes of the network. RNNs include both feedforward structures and feedback connections, where the structures that include feedback connections provide context to the network. The feedback connections reach from the hidden layers or the output layers to nodes that are called context nodes, which are located in the preceding layers of the network [41]. This early modeling of the RNN has since been improved on, with the context length becoming theoretically infinite, with the drawback of extremely slow training times [14]. The ability to remember is what makes RNNs particularly suited for detecting patterns in sequential and time-varying data, allowing them to have a breadth of applications ranging from NLP to computer vision to signal processing [41]. RNNs present their own limitations with long training times, complicated backpropagation algorithms, and vanishing gradients, making them less than optimal for long-term dependencies. RNNs can be partially connected or fully interconnected. The first RNN was introduced by J.J. Hopfield in 1982 in his paper Neural networks and physical systems with emergent collective computational abilities [28].

2.4.1 Architecture of Recurrent Neural Networks

In comparison to FNNs, when training an RNN there is no need to select context size, as the whole context size can theoretically be used. Context size refers to the number of words that are taken into account in the model's decision-making. This is a significant improvement over the FNN, as configuring the size of the context is often a very tedious task [14]. However, the full, infinite context can't be used due to the exploding and vanishing gradient problem. Methods such as Long Short-Term Memory Units (LSTMs) have been developed to combat the vanishing gradient issue. Moreover, the RNN pales when it comes to interpretability, whereas FNNs are more intuitive [14]. Interpretability is an issue that is increasingly important within the field of NLP due to recent developments in LLMs that have sparked discussions about Artificial General Intelligence (AGI), an unrealized concept of AI models that possess human-like ability to learn. A traditional RNN architecture, which has been unfolded into an FNN is as follows [49]:

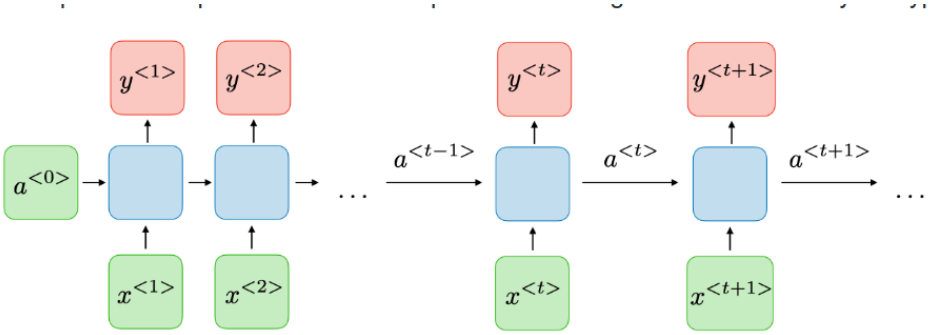

Figure 2.2: General many-to-many RNN architecture [49]

where $x^{<i>}$ is an input in an input sequence, $a^{<i>}$ is an activation function and $y^{<i>}$ is an output of the RNN for $x^{<i>}$. This is a representation of an RNN of type many-to-many. In a many-to-many architecture, sequences are fed into the networks one at a time, with the network always receiving the previous output $y^{<i>}$ as the input with a possibility of another member of the input sequence being fed in subsequently, which ultimately produces the output $y^{<i+1>}$. This iterative process is repeated until the last member of the input sequence has been processed [33]. One iteration is commonly called a timestep t. A timestep t corresponds to the state of the network after the input $x^{<t-1>}$. For example, after the first input in a sequence, $t = 1$. Other architectures include many-to-one, where outputs are not generated from individual members of the input sequence, but the hidden state is passed on to the next timestep of the network. Many-to-one RNNs

are commonly used for tasks such as sentiment analysis or text classification. One-to-many, where multiple outputs are generated from a sequence of length 1, the input might represent a genre of music, and the output the notes of a generated song. One-to-one, which corresponds to a traditional FNN [33].

As Deep Learning (DL) models, simple RNNs are composed of three general components, the input layer, hidden layers with hidden recurrent states, and the output layer. The major difference to FNNs is the presence of recurrent hidden states. As is the case with the Perceptron, the input of the neural network can be represented as a vector $\vec{x}$ that's cells represent the features of the input data, where the size of the vector $\vec{x}$ is equal to the number of neurons in the input layer. In RNNs, a popular choice for the activation function is the hyperbolic tangent, which is defined as

$$tanh = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

[51], but similarly to the Perceptron, ReLU, and Sigmoid are common activation functions. RNNs are optimal for sequential data, and there exist many different ways to map inputs to outputs. Different mappings correspond to different kinds of architectures of RNNs.

For each RNN, there exists an equivalent FNN that has a finite number of steps [51]. The FNN can be found by unwrapping the RNN to parts such as the ones presented in figure 2.2. These unrolled RNNs can be trained with a special case of backpropagation called Backpropagation Through Time (BPTT) [51]. This also removed the initial limitation of backpropagation not being suitable for training RNNs. BPTT is a supervised training algorithm applied in the same way as the general backpropagation in conjunction with gradient descent. Backpropagation is not without issues when it comes to RNNs. For backpropagation to be used, the RNN must be unwrapped into a sequence of FNNs that share the same set of weights with each other. Each new FNN receives a new input and is linked to the previous FNN via activation functions. As the gradients of the activation functions propagate over their predecessors, they multiply each other. Due to the weights being shared across the FNNs, the multiplications happen between weights that are likely to be of a similar value. This leads to the gradient either fading to obscurity with small weights or exploding up to very high values [33], thus making it difficult for the RNN to learn long-term dependencies.

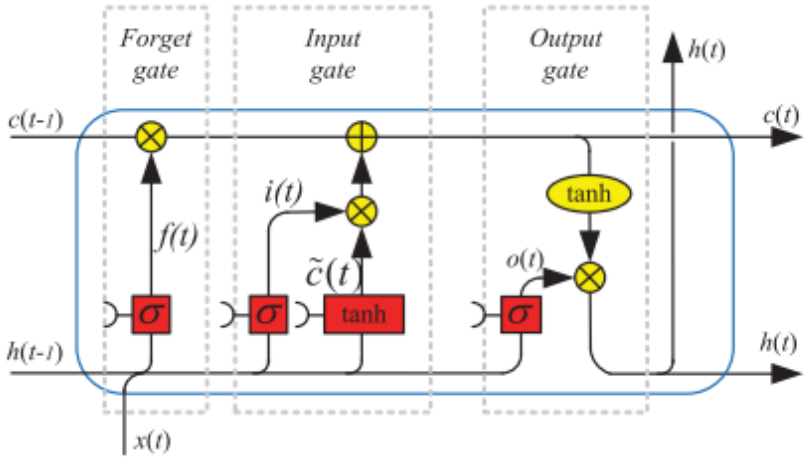

Figure 2.3: Architecture of LSTM with a forget gate [61].

2.4.2 LSTM

Long Short-Term Memory was introduced by Hochreiter and Schmidhuber in 1997 [27] in their attempt to deal with the long-term dependency issue in RNNs. LSTMs handle the problem of vanishing and exploding gradients by introducing gate functions into the structure of the neurons. In the most barebones implementation of LSTM, input and output gates operate within the cell to decide *whether to retain a specific input in the cell state*, respectively the output gate decides *what kind of information can be output from the cell* based on the cell state [61]. However, when LSTM cells are generally in reference to cells that in addition to the input and output gates, have an additional forget gate preceding the input gate. The forget gate can decide whether the cell state completely discards certain information from the cell state. The cell state represents the long-term memory of the LSTM and is not present in RNNs.

LSTM architecture with a forget gate can be examined in Figure 2.3, where $x(t)$ denotes the input, $h(t)$ the recurrent information, $c(t)$ denotes the cell state, at timestep t. The forget-gated LSTM is not without its issues. There are no direct connections between the cell state and the gates, which leads to loss of information and thus harms the model's performance. For this, there is another architecture that utilizes peephole connections to establish information flow between the cell's current internal states. This approach allowed the LSTM to learn to time, count, and act in a nonlinear fashion, which in turn makes the LSTM a compelling model for real-world use cases.

After its introduction, LSTMs became the center of attention in DL research and they are

still used today in a variety of tasks involving sequential data. In [61] LSTMs are divided into two categories; LSTM-dominated networks and integrated LSTM networks [22].

2.5 Convolutional Neural Networks

A Convolutional Neural Network, also known as ConvNet, is an Artificial Neural Network. CNN architecture dates back to 1979 when Kunihiko Fukushima introduced the Neocognitron, a model utilizing convolutional layers [19]. The defining property of a CNN is the use of convolutional layers, which themselves are composed of filters that can detect patterns in the input data. This makes the CNN intuitively fitting for image recognition, which remains the primary use for CNNs. Still, CNNs can be utilized in NLP and sequential data and have proved to be the natural choice when wanting to extract higher-level features from n-grams or constituting words [60]. CNNs have considerable limitations. They require a particularly large data set to efficiently train the huge number of trainable parameters. Additionally, long-term dependencies are not retained, which makes CNNs sub-optimal in determining the effect of mutually distant tokens [60].

2.5.1 Architecture of Convolutional Neural Networks

CNNs take advantage of a myriad of different techniques when it comes to architecture. They combine fully connected layers with convolutional layers and pooling layers. CNNs can be trained with backpropagation. Convolutional layers can learn feature representations from the input data, which it does with so-called convolution kernels. The purpose of a convolution kernel is to compute a feature map, which in turn can be used for predictions. Convolution kernels in shallower layers of the network are designed to detect low-level features such as edges or curves, whereas kernels deeper into the network can detect more abstract features. The final feature maps are constructed from outputs of multiple kernels [24].

Pooling layers are typically inserted between convolutional layers to reduce the resolution of the feature maps. This reduces both shift invariance and the connections needed between the convolutional layers. Shift invariance refers to the output of the model being indifferent to shifts in the order of the input data. CNNs were long thought to be shift-invariant due to convolutional layers being shift equivariant and pooling layers, which build stability to deformations [9]. However, CNNs have been found to be affected with a

probability as large as 30 percent by a mere 1-pixel shift in the input image. This problem has been addressed by replacing conventional linear pooling layers with adaptive polyphase sampling, which has proved to make models that utilize it 100 percent shift invariant. In layman's terms, shift invariance simply allows the CNN to recognize features that aren't exactly like the ones it has been trained on, thus improving generalizability. There are different kinds of methods for pooling, which include max pooling, average pooling, and min pooling.

After the convolutional, pooling, and feedforward reasoning layers, there is most often a softmax output layer. The softmax layer is popular due to its fit for classification tasks. Another approach is to use SVM in conjunction with the features of the CNN to help in classification tasks [24]. A great case study for the architecture of a CNN is the famous LeNet-5, which was one of the earliest convolutional neural networks.

2.6 Pre-Transformer State-Of-The-Art

In machine learning tasks, model performance is commonly measured by three different measurements called accuracy, recall, and F1-score. Accuracy can be thought of as the quality of the predictions that the model is making, as in what percentage of the positive predictions was actually correct. Recall measures the proportion of positive samples that were predicted correctly. F1-score is slightly more sophisticated and it is measured as the harmonic mean of precision and recall. F1-score is described as the accuracy of the model. The measurements are based on the distribution of true and false predictions. In binary classification, a prediction can be one of four types, True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). The following are the mathematical definitions of precision, recall, and F1-score:

$$Precision = \frac{TP}{TP\ +\ FP} \qquad Recall = \frac{TP}{TP\ +\ FN} \qquad F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Additionally, other measurements like Mean Reciprocal Rank (MRR) and Mean Average Precision are used by [59]. MRR and MAP are better suited for evaluating tasks where a list of possible responses is produced based on probability.

Pre-Transformer state-of-the-art refers to model performance in NLP tasks before Transformer-based models started to appear in late 2017 and after. In 2017 and the few preceding years, the state-of-the-art multitasking models combined techniques from different domains. One

such model is introduced by Kaiser et al. [31] and takes advantage of convolutional layers, attention mechanism, and sparsely-gated layers to reach good performance in various language tasks like image captioning, classification, and language translation. Despite good performance, the model presented by [31] was not able to reach state-of-the-art when compared to task-specific models. In terms of state-of-the-art task-specific models, rapid competition existed between RNNs like LSTMs, Gated Recurrent Units (GRUs), and CNNs. Despite NLP tasks being sequential in nature, CNNs were continuously able to outperform past RNN models [59], and vice versa.

The architectures were compared systematically in 2017 by Yin et al. [59] in the following tasks: sentiment/relation classification, textual entailment, answer selection, question-relation matching in Freebase, Freebase path query answering, and part-of-speech tagging. The Stanford Sentiment Treebank (SST) and WikiQA datasets are used in [59] to measure CNN's, LSTM's, and GRU's performance in sentiment classification and question answering. The SST is a fully labeled parse tree dataset that consists of movie reviews and their sentiment, in which the sentiment is either positive or negative. In SST, the best performance was derived from the GRU model, which reaches an 86.32 percent accuracy, leading the LSTM by up to 1.8 percent and the CNN by up to 3.9 percent. A more interesting category that LLMs are tested on is question answering. In the tests conducted, the WikiQA dataset was used to evaluate model performance by MRR. The WikiQA dataset is a collection of question and sentence pairs in which there may be 0 or more correct sentences corresponding to a question [57]. The results in Answer Selection with the WikiQA dataset revealed the CNN to be superior to the RNN-based models by an advantage of up to 1.4 points.

Machine Translation (MT) is another significant task of LLMs. MT is commonly evaluated using a collection of datasets from Workshop on Statistical Machine Translation (WMT). It consists of four different tasks; News translation, quality estimation, metrics, and medical translation. Machine translation is evaluated on an algorithm called the Bi-Lingual Evaluation Understudy (BLEU), in which the measure of quality is how closely the machine translation mimics the human-translated equivalent. As such, BLEU is a metric of similarity between 0 and 1. State-of-the-art results prior to the Transformer for the WMT2014 English-French dataset measured in BLEU were 41.44 [20], achieved with a fully convolutional sequence-to-sequence model called ConvS2S. ConvS2S was also able to produce state-of-the-art results in WMT2014 English-German and WMT2016 English-Romanian datasets, triumphing the previous records by 0.5 BLEU and 1.9 BLEU respectively.

2.7 Bottlenecks of NLP Before the Transformer

Despite different gating methods used by LSTMs and GRUs, long-term dependencies continued to be the bottleneck in NLP applications. While LSTMs improved the long-term dependency issue in RNNs and made the context window theoretically infinite, applying such a long context size in practice is not possible. The average context size of an LSTM model is only around 200 tokens [34]. Moreover, successful LSTMs utilizing caching rely heavily on the cache for long-term dependencies and do not distinguish between the word order of long-term dependencies that are outside a 50 token range[34]. This limited the ability of state-of-the-art models significantly, as global context was not available to the model. In addition to that, some tokens are entirely ignored by RNN-based models, resulting in a loss of information.

Parallelization before the Transformer model was a very prevalent issue, notably with RNN-based models, as their training is fundamentally a sequential operation. Nevertheless, some parallelization techniques exist for RNNs. Some of these include asynchronous training algorithms that reduce synchronization overhead between distributed neuron nodes and the interactive node via sampling and mean fusion strategies. Without exploring this in detail, these strategies were found to enhance training efficiency while only decreasing model accuracy by less than 1 percent on average [45]. [45] is notable, as it takes an alternative approach to the more common parallelization techniques called model parallelization and data parallelization. Data Parallelism divides the dataset used for training into smaller batches that are then used over multiple GPUs to train the multiple copies of the model in parallel. This is done because the size of the data may exceed the available memory of the GPU being used. At the end of each training step, the models can be synchronized into one unified model. Another approach is to not synchronize the models, but instead, carry out the predictions of the final model by performing a vote between each of the separate models. Model parallelism is similar to data parallelism in principle, but instead of splitting the data, it splits the model into smaller components that fit onto the memory of a single GPU, thus addressing the case of very large models, such as large pre-trained transformers. Data and model parallelization introduce notable obstacles for RNNs, as both of the techniques require the synchronization of gradients, making the training inefficient [45]. Parallelization is less of an issue for CNNs, but CNNs lack the memory of RNN-based models. With efficient parallelization out of the question for RNNs, training duration blows out of proportion for large datasets. Model and Data

parallelism can much better be used with models like the Transformer, as a single input must only be processed once, with no recursion needed.

The Transformer model rids itself of the issues discussed here, as it is able to both remember long-term dependencies and be trained with parallelization techniques hard for RNN-based models. The Transformer model architecture is introduced and detailed in the 2017 paper [56] and is summarized in the next section.

3 The Transformer Model

The Transformer is a neural network model architecture that applies encoder-decoder architecture in conjunction with self-attention, multi-head attention, point-wise feedforward networks, and positional encoding to carry out sequence transduction among many other use cases. Its key innovation is the use of self-attention, which relates the positions of a given sequence to compute its representation. Another distinguishing factor of the Transformer is the lack of recurrent or convolutional layers, which enables it to be trained with higher levels of parallelization without losing memory. Together, these attributes make the Transformer the superior choice in a vast array of different NLP tasks, such as machine translation, sentiment analysis, textual entailment, text summarization, and generation.

3.1 Architecture of The Transformer Model

The architecture of the Transformer is described in a flowchart in Figure 3.1. Like most other competitive approaches to MT, the Transformer consists of an encoder-decoder structure. The encoder's job is to map a sequence of symbol (e.g. word) representations to an equal-length sequence of continuous representations. The continuous representations encode as much information as possible about the given symbols, which includes the position of the symbol in the sequence and the semantic interpretation. The continuous representations are referred to as word embeddings. The word embeddings are then fed into the decoder along with the outputs that may have already been generated by the Transformer, this is known as auto-regression [56]. The decoder then generates a new sequence of symbols one symbol at a time based on the inputs and the outputs generated so far. The process is repeated until an "end" token is generated.

3.2 The Journey Through The Transformer

To demonstrate and learn the architecture of the Transformer, let's examine a text inputs journey through the encoder-decoder-based architecture and briefly explain the techniques used as we go. Let there be a text input *Who are you?* that receives an output *I am a bot.* generated by the Transformer.

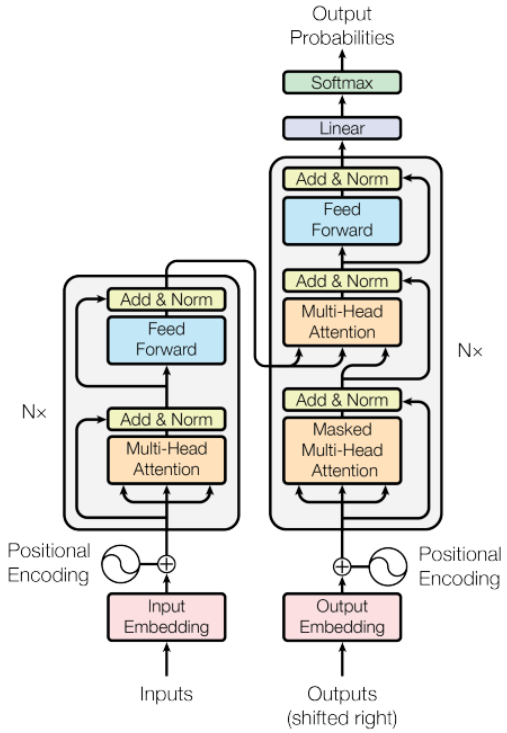

Figure 3.1: Transformer model architecture [56]

3.2.1 Word embeddings and Positional Encoding

As the symbols are fed into the Transformer, they must be encoded into the aforementioned word embeddings. The original Transformer model utilized what is called learned embeddings, in which the words are represented as vectors of length d_{model} with similar words existing close to one another in the d_{model}-dimensional vector space. In our example, the input tokens Who, are, and you? would be encoded into vectors of dimension d_{model}. After the word embedding has been looked up, a positional encoding is calculated based on the index of the word in the sequence. A positional encoding communicates to the model the relative index of the word in the given sequence and is of the same dimension as the word embedding. There are multiple options as to how a positional encoding can be applied to an embedding. In [56], an interesting algorithm based on the sine and cosine functions is used. In their implementation, each dimension of the positional encoding corresponds to a sinusoid. The algorithm for the positional encoding vector is the following:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}) \qquad PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

where *pos* is the position and i is the dimension. A sum operation is then performed between the positional encoding and the word embedding to form the final product of the embedding layer.

3.2.2 The Encoder Stack

The word representations are then fed into the encoder stack, which in the original architecture consists of 6 identical layers, each containing 2 sub-layers, the multi-head self-attention sub-layer, and the position-wise fully connected feedforward layer. Each of the sub-layers accepts as input and produces as output d_{model}-dimensional data.

3.2.3 Multi-head Self-attention

The multi-head self-attention is one of the most important components in the Transformer, as it replaces the convolutional and recurrent layers. Multi-head attention contains multiple "heads" of self-attention. Self-attention is a technique used to relate tokens of the input sequence to each other. This way the model can recognize whether a previous input word affects the current word in some way. For this, the Transformer uses an attention mechanism called Scaled Dot-Product Attention (SDPA). The following is a description of the self-attention process, which is performed in parallel for vectors named Query Q, Key K, and Value V. Each head of self-attention receives as input a slice of each Q, K, V vector. The Q, K, and V vectors are derived by feeding the word embedding to three unique FNNs.

SDPA receives as input three vectors that are derived from the word embedding by feeding it into three separate fully connected layers. The dot products of the query are computed for each key. The result of this is called a score matrix, which determines the relation between the words in the input sequence. The score matrix is further scaled by $\frac{1}{\sqrt{d_k}}$, where d_k is the dimension of the query and key vectors. The scaling is done to prevent exploding values, as multiplication is known to cause issues with scale. The result of the scaling is fed into a softmax layer to obtain the weights for the value vector V. The softmax produces a matrix of probabilities for each relation of each word. Finally, a matrix multiplication is performed between the value vectors and the output matrix of the softmax. This leads to irrelevant words being multiplied by smaller probabilities than the relevant ones, which leads to the model paying more attention to the relevant words. Further processing is applied by feeding the resulting vector into a linear fully connected

layer before being passed on to the second sub-layer of the encoder.

3.2.4 Residual Connection and Normalization

A residual connection is a path or a shortcut for input to skip over certain layers in the architecture. A residual connection is used to combine the output of the multi-headed attention layer with its input, as it has been found to improve the accuracy of the model and make the model easier to optimize [26]. The summation is then normalized with [2]'s method to reduce training time [56].

3.2.5 Position-wise Fully Connected Feedforward Network

The encoder layer includes a position-wise fully connected FNN with ReLU activation in between. While the FNN is identical for each position, they use different parameters across the layers [56]. Similarly to the multi-head step, a residual connection is used to combine the output of the FNN with its input, and the summation is normalized before being fed to the decoder.

3.2.6 The Decoder Stack

The decoder's task is to generate sequence outputs. The Transformer outputs one token at a time, and the decoder stack receives each of the previous outputs as an additional input sequence in addition to the current input in order to predict the next output of the sequence. Each of the three sub-layers in the decoder stack employ a residual connection, which is used for performing a sum operation with the input and the output of the sub-layer.

3.2.7 Decoder Multi-head Self-attention and Masking

Receiving the previously generated outputs as additional input is called auto-regression. Similarly to the initial input, a word embedding with positional encoding is generated for the previous outputs. Before concatenating the previous output with the new input, it is fed through a masked multi-head attention layer. Auto-regression introduces a problem where the model relates past words to previous outputs in the score matrix. In order to keep the auto-regressive property of the Transformer, the decoder's multi-head layer

implements something called masking. In masking, illegal relationships are marked as negative infinity in the score matrix, thus making the model ignore them completely. Identically to the multi-head attention in the encoder, the output of the masked multi-head attention is summed with the residual connection and normalized.

The normalized output is then used as the Value vector V in the following layer of the decoder, which is another multi-head attention layer. The Query Q and Key K vectors come from the encoder's output. This combines the previous outputs with the current input of the model.

3.2.8 Feedforward Neural Network, Linear Classification, and Final Token Probabilities

The final layer of the decoder is another FNN, which is identical to the previous FNNs in the model. A learned linear transformation that outputs a vector of size n is performed on the final output of the decoder before passing it into a softmax layer, which will produce the final probabilities for the next token [56]. For example, if the library of tokens consists of n words, the output of the last layer is a vector of length n. The index of the greatest scalar in the vector corresponds to the index of the next word. The predicted word is then appended to the list of previous inputs and fed into the decoder. The process is repeated until the model produces an end token.

3.3 Limitations of The Transformer Model

The Transformer has limitations. Its time complexity is quadratic due to the Scaled Dot-Product Attention mechanism. This makes the Transformer slow in processing lengthy inputs [32]. This, however, has been addressed by multiple papers such as [13].

Another perhaps surprising limitation of the Transformer is that of its heart, the Scaled Dot-Product Attention. According to [25], self-attention is fundamentally limited in its computational power. Self-attention can't model basic recursion or periodic regular languages. From this, it can be derived that even when infinite precision is allowed for the Transformer, it still can't emulate general finite-state automata like RNNs can. This affects the Transformer's performance with long inputs, making mistakes inevitable if the amount of heads in multi-head attention is not increased together with the input length. It is noted in the paper, however, that the success of the Transformer indicates that such

computational complexity may not be required for language modeling. For instance, language uses recursion only in very restricted ways due to cognitive factors and is thus not very important for self-attention to model.

Some more practical limitations of the Transformer include the need for a large corpus of training data, its sequential and ambiguous nature, and limited interpretability as a deep learning-based model.

4 The Transformer Model's Impact

The Transformer model has had a transformative impact on NLP. The performance improvement brought on by effective parallelization has enabled the training of models of colossal sizes. A lengthy context window and efficient mapping of token relations has improved the accuracy and coherence of language models to represent more human-like text. The vast majority of the most important benchmarks such as the WMT tasks are dominated by transformer-based models. A previously under-utilized method of pre-training and fine-tuning has brought the benefits of the best-performing and most world-knowledgeable models to everyday developers, resulting in an influx of new ventures in the space. Furthermore, the utilization of human feedback in training bleeding-edge models and solutions such as flash attention [13] have addressed the most pressing issues of the Transformer.

4.1 Overview of The Tasks Transformers Achieve High Performance In

4.1.1 The Original Transformer Model

The results of the original Transformer model proposed by Vaswani et al. are described in [56]. They are limited to MT tasks from the WMT 2014 datasets with two versions of the model being used to make translations. The versions are called *base model* and *big model*. The big model performs at a higher level than the base model, but the base model can still generate state-of-the-art results. The models were trained using NVIDIA's P100 GPUs. The big model was trained for 3.5 days and the base model was trained for around 12 hours [56].

WMT 2014 Translation Tasks

The big Transformer model proposed by Vaswani et al. performed better than all the previous state-of-the-art models on the English-to-German translation task while using significantly less computation time and power. The new state-of-the-art established by

the Transformer model at the time of the paper's release was 28.4 BLEU. That is an improvement of more than 2.0 BLEU with significantly reduced training time. To measure the effect of different components of the architecture, Vaswani et al. "varied their base model in different ways". The changes were then measured by the changes in performance on the English-to-German translation task with its development set called newstest2013 [56]. The tests concluded that in multi-head attention, more heads are better than just one, but too many heads will decrease performance; decreasing the attention key size hurts the model's performance; and larger models, as in more trainable parameters, result in increased performance.

The new state-of-the-art result established by the Transformer model in English-to-French is 41.0 BLEU, some 0.44 increase in BLEU for the top performing single model MoE while only using around one-fourth of the training cost.

4.1.2 Transformer-based Models

The revolution of NLP begins with the Transformer model, but the real advances are not made by the Transformer model in the form Vaswani et al. introduced it. They are made with models that rely on the architecture of the Transformer, making it better with modifications. These models are called Transformer-based models (alternatively X-formers) in the literature. Some of the most notable Transformer-based models are Google's BERT [15] and its variants like RoBERTa [39], ALBERT [36], ELECTRA [11], and OpenAI's GPT models. Transformer-based models dominate the benchmarks on the vast majority of NLP tasks. Let's take a look at the following tasks: Text Classification (TC), Question Answering (QA), and Machine Translation (MT). These tasks have been chosen as they represent classification, text generation, and translation tasks, which demonstrate the diversity of NLP tasks.

Text Classification

TC is a classic NLP task in which a text is presented to the model for it to predict the category that the text is a part of. The IMDb Benchmark dataset has gained notoriety as a landmark text classification dataset. Countless different models have been applied to derive sentiment from the IMDb reviews, these include GRUs [53], SVMs, Naïve Bayes, among others [55]. At present, transformer-based models reign the leaderboards with a model called Transformer-XL, and its variants XLNet and ERNIE-Doc being the state-of-

the-art [58]. ERNIE-Doc-Large achieves the best performance, scoring an impressive 97.1 accuracy on the classification task [16].

Question Answering

QA is a field of natural language processing that has experienced a total transformation after the Transformer model was introduced in 2017. Long-term dependencies are extremely important in generating responses to questions in which the answer is highly context-dependent. The top-performing model at the present time is a model called ANNA [30], which is a model specifically tailored for QA tasks. Anna reaches an F1-score of 95.719, XLNet follows closely behind with an F1-score of 95.080 [58].

Machine Translation

MT is another ubiquitous benchmark for the assessment of LLMs. It is also another area that Transformer-based models reign supreme. Here, we can yet again take a look at the renowned WMT2014 datasets, such as English-to-German. All 5 top-performing models are Transformer-based models, with the best results being generated by a novel technique called parameter sharing for Transformers [52].

4.1.3 Performance Conclusion

The state-of-the-art metrics are dominated by Transformer-based models with record metrics published yearly. To illustrate the dominance of the Transformer, examine this table of record holders in popular tasks and datasets.

Dataset	IMDb		SQuAD1.1		WMT2014 E-t-G	
Task / Metric	TC	Accuracy	QA	F1	MT	BLEU
#1	ERNIE-Doc-Large	97.1	ANNA	95.72	Transformer Cycle	35.14
#2	XLNet	96.8	LUKE	95.38	back-translations	35.00
#3	RoBERTa	96.6	XLNet	95.08	Transformer+Rep	33.89

4.2 Byproducts of The Transformer Model

The Transformer model's robustness paired with swift innovation from affluent industry players is enabling efficient new ways to work with LLMs. It is also strengthening their pre-existing use cases. The following are areas of NLP that have experienced an overhaul with the coming of the Transformer model.

4.2.1 Pre-training

Pre-training means creating a foundation model by training a chosen model with typically a large corpus of sequential data. The pre-trained model can then be fine-tuned, which translates to training the pre-trained model with yet more data for a specific task. Pre-training is known to achieve significant performance gains in various tasks. In addition, when pre-training the model with seemingly unrelated data with respect to the final task, the model gains world knowledge, which can broaden the model's understanding of the final task.

While Pre-trained neural networks existed before the Transformer model, they were not as popular as they have become due to the utility they provide for Transformers. Since then, massive effort has been dedicated to pre-training Transformer models on large-scale text corpora, which is believed to be one of the major reasons of the Tranformer's wide application in NLP [37]. A commercial landmark paper on pre-training, and more specifically generative pre-training is [48], which demonstrates that large gains on various NLP tasks can be realized by generative pre-training of language models. Due to the robustness of Transformers and the advances in computation, pre-training can be done with an extensive corpus without compromising model performance.

4.2.2 Generative AI

The Transformer model's significance in the popularization of generative AI is undisputed. Generative AI is among the first signs of AI breaking into mainstream consumer products as the main attraction. Transformers in generative AI have proved themselves mainly in the context of text generation. In the public eye, commercial LLMs like the GPT family have completely transformed the landscape of NLP, which has been a major contributor in generative AI companies raising up to 1.6 billion in funding in the first quarter of 2023 [21].

Other pre-trained models include the BERT model and its variants. Generative AI requires vast amounts of data, which Transformers are exceptionally good with in comparison to previous state-of-the-art models like LSTMs and RNNs. This is why the architecture has allowed for colossal models such as GPT-4, which boasts 175 billion trainable parameters [7], 10 times more than any previous non-sparse LM.

4.2.3 Multi-lingual and Multi-modal Capabilities

Pre-trained transformers have demonstrated success in generalizing across languages [35]. multilingual BERT [15], for example, was trained on 104 languages and when fine-tuned for new downstream tasks, it can transfer resources from resource-rich languages like English to low-resource languages like Swahili. This essentially means that knowledge is possible to have knowledge transferred from one language to another. Thus, Transformers have opened new avenues for multilingual language processing.

Multi-modal inputs are another advancement that has sprung from the Transformer. Recently, models like GPT-4 have demonstrated impressive capabilities in understanding multi-modal content that includes images and text. The text and images can be "arbitrarily interlaced" [46]. Embedding images into the input data has not degraded the quality of the outputs according to [46]. Another notable multi-modal LM is ViLBERT, which is a variation of BERT that has been extended to a multi-modal two-stream model to process visual and textual inputs. The inputs are given separately and concatenated in so-called co-attentional layers of the Transformer-based model's architecture. At the time of its release, ViLBERT could have been considered a state-of-the-art model in its class, reaching state-of-the-art on all 4 tasks it was proposed with [40].

5 Conclusion

Transformer-based models are improving constantly with more papers released on novel ways to overcome the foundational issue of the original Transformer model, which is the quadratic computational complexity of the self-attention mechanism with respect to the input sequence length [54]. With the attention and funding that machine learning and AI have attracted in light of the recent commercial breakthroughs of NLP systems, it can be speculated that the rate of innovation will only accelerate in NLP with transformer-based solutions. The Transformer model has been able to vitalize the field of NLP with its qualities such as parallelization, long-term dependency capabilities, and multi-modality. It is especially interesting to see the already-present transformation that rapid innovation will bring in the following decades.

This thesis was set to uncover the history of NLP, the rise of deep learning, what the Transformer architecture is, and ultimately how it has impacted the field. Broadly, the Transformer has brought on a new era in AI research and utilization, embarked on the commercial revolution of AI, and solidified paradigms like pre-trained machine learning models and generative AI. Transformer-based models are vastly the architecture of choice when approaching any given NLP task, and have populated the research area with state-of-the-art solutions as is illustrated in section 4.1.3. Massive, multi-modal, and multi-lingual models have been proposed. Transformers are now the industry standard for commercial NLP applications, bridging the gap between high and low-resource languages, and are actively becoming more general with multi-modality [8]. I hope this thesis has helped the reader in getting a hold of the disorderly world of NLP through a historical overview, the Transformer model and its components, the relevant benchmarks, and the emerging byproducts of NLP innovation like generative NLP, multi-modality, and commercialization.

Bibliography

[1] D. Andina, A. Vega-Corona, J. Seijas, and J. Torres-Garcıa. “Neural networks historical review”. In: *Computational Intelligence: for Engineering and Manufacturing* (2007), pp. 39–65.

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton. “Layer normalization”. In: *arXiv preprint arXiv:1607.06450* (2016).

[3] D. Bahdanau, K. Cho, and Y. Bengio. “Neural machine translation by jointly learning to align and translate”. English. In: International Conference on Learning Representations, ICLR, 2015.

[4] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. “Automatic differentiation in machine learning: a survey”. In: *Journal of Marchine Learning Research* 18 (2018), pp. 1–43.

[5] Y. Bengio, R. Ducharme, and P. Vincent. “A neural probabilistic language model”. In: *Advances in neural information processing systems* 13 (2000).

[6] O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, and A. Tamchyna. “Findings of the 2014 Workshop on Statistical Machine Translation”. In: *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Baltimore, Maryland, USA: Association for Computational Linguistics, June 2014, pp. 12–58. DOI: 10.3115/v1/W14-3302.

[7] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[8] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang. *Sparks of Artificial General Intelligence: Early experiments with GPT-4*. 2023. arXiv: 2303.12712 [cs.CL].

[9] A. Chaman and I. Dokmanic. "Truly shift-invariant convolutional neural networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 3773–3783.

[10] K. R. Chowdhary. "Natural Language Processing". In: *Fundamentals of Artificial Intelligence*. New Delhi: Springer India, 2020, pp. 603–649.

[11] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators". In: *International Conference on Learning Representations*. 2020.

[12] R. Collobert and J. Weston. "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning". In: ICML '08. Association for Computing Machinery, 2008, pp. 160–167. DOI: 10.1145/1390156.1390177.

[13] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré. *FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness*. 2022. arXiv: 2205.14135 [cs.LG].

[14] W. De Mulder, S. Bethard, and M.-F. Moens. "A survey on the application of recurrent neural networks to statistical language modeling". In: *Computer Speech Language* 30.1 (2015), pp. 61–98. DOI: https://doi.org/10.1016/j.csl.2014.09.005.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "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)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[16] S. Ding, J. Shang, S. Wang, Y. Sun, H. Tian, H. Wu, and H. Wang. "ERNIE-Doc: A Retrospective Long-Document Modeling Transformer". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Aug. 2021, pp. 2914–2927. DOI: 10.18653/v1/2021.acl-long.227.

[17] H. L. Dreyfus. “From micro-worlds to knowledge representation: AI at an impasse”. In: *Mind design* (1981), pp. 161–204.

[18] C. Dyer, A. Kuncoro, M. Ballesteros, and N. A. Smith. “Recurrent Neural Network Grammars”. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 199–209. DOI: 10.18653/v1/N16-1024.

[19] K. Fukushima. “Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. en. In: *Biol. Cybern.* 36.4 (1980), pp. 193–202.

[20] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. “Convolutional sequence to sequence learning”. In: *International conference on machine learning*. PMLR. 2017, pp. 1243–1252.

[21] *Generative AI startups jockey for VC dollars*. Accessed: 2010-09-30.

[22] F. A. Gers and J. Schmidhuber. “Recurrent nets that time and count”. In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. Vol. 3. IEEE. 2000, pp. 189–194.

[23] Y. Goldberg. “A primer on neural network models for natural language processing”. In: *Journal of Artificial Intelligence Research* 57 (2016), pp. 345–420.

[24] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen. “Recent advances in convolutional neural networks”. In: *Pattern Recognition* 77 (2018), pp. 354–377. ISSN: 0031-3203.

[25] M. Hahn. “Theoretical Limitations of Self-Attention in Neural Sequence Models”. In: *Transactions of the Association for Computational Linguistics* 8 (Jan. 2020), pp. 156–171. DOI: 10.1162/tacl_a_00306.

[26] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[27] S. Hochreiter and J. Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[28] J. J. Hopfield. “Neural networks and physical systems with emergent collective computational abilities.” In: *Proceedings of the National Academy of Sciences* 79.8 (1982), pp. 2554–2558. DOI: 10.1073/pnas.79.8.2554.

[29] K. S. Jones. “Natural language processing: a historical review”. In: *Current issues in computational linguistics: in honour of Don Walker* (1994), pp. 3–16.

[30] C. Jun, H. Jang, M. Sim, H. Kim, J. Choi, K. Min, and K. Bae. “ANNA: Enhanced Language Representation for Question Answering”. In: *Proceedings of the 7th Workshop on Representation Learning for NLP*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 121–132. DOI: 10.18653/v1/2022.repl4nlp-1.13.

[31] L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit. “One model to learn them all”. In: *arXiv preprint arXiv:1706.05137* (2017).

[32] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. “Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by H. D. III and A. Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 5156–5165.

[33] M. Kaur and A. Mohta. “A Review of Deep Learning with Recurrent Neural Network”. In: *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*. 2019, pp. 460–465. DOI: 10.1109/ICSSIT46314.2019.8987837.

[34] U. Khandelwal, H. He, P. Qi, and D. Jurafsky. “Sharp Nearby, Fuzzy Far Away: How Neural Language Models Use Context”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 284–294. DOI: 10.18653/v1/P18-1027.

[35] Y. Khemchandani, S. Mehtani, V. Patil, A. Awasthi, P. Talukdar, and S. Sarawagi. “Exploiting Language Relatedness for Low Web-Resource Language Model Adaptation: An Indic Languages Study”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 1312–1323. DOI: 10.18653/v1/2021.acl-long.105.

[36] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. "ALBERT: A LITE BERT FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS". In: *8th International Conference on Learning Representations, ICLR 2020*.

[37] T. Lin, Y. Wang, X. Liu, and X. Qiu. "A survey of transformers". In: *AI Open* (2022).

[38] S. Linnainmaa. "The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors". PhD thesis. Master's Thesis (in Finnish), Univ. Helsinki, 1970.

[39] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL].

[40] J. Lu, D. Batra, D. Parikh, and S. Lee. "ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019.

[41] L. R. Medsker and L. Jain. "Recurrent neural networks". In: *Design and Applications* 5 (2001), pp. 64–67.

[42] T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient estimation of word representations in vector space". In: *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*. 2013.

[43] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. "Distributed representations ofwords and phrases and their compositionality". In: 2013.

[44] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman. "Natural language processing: an introduction". In: *Journal of the American Medical Informatics Association* 18.5 (Sept. 2011), pp. 544–551. ISSN: 1067-5027. DOI: 10.1136/amiajnl-2011-000464.

[45] D. Niu, T. Liu, T. Cai, and S. Zhou. "The Asynchronous Training Algorithm Based on Sampling and Mean Fusion for Distributed RNN". In: *IEEE Access* 8 (2020), pp. 62439–62447. DOI: 10.1109/ACCESS.2019.2939851.

[46] OpenAI. *GPT-4 Technical Report*. 2023.

[47] B. Pang, L. Lee, and S. Vaithyanathan. “Thumbs up? Sentiment Classification using Machine Learning Techniques”. In: *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing, EMNLP 2002*. 2002, pp. 79–86.

[48] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. *Improving language understanding by generative pre-training*. 2018.

[49] *Recurrent Neural Networks cheatsheet*. https://stanford.edu/ shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks. Accessed: 2023-04-03.

[50] F. Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.

[51] A. Sherstinsky. “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network”. In: *Physica D: Nonlinear Phenomena* 404 (2020), p. 132306. ISSN: 0167-2789. DOI: https://doi.org/10.1016/j.physd.2019.132306.

[52] S. Takase and S. Kiyono. “Lessons on parameter sharing across layers in transformers”. In: *arXiv preprint arXiv:2104.06022* (2021).

[53] D. Tang, B. Qin, and T. Liu. “Document Modeling with Gated Recurrent Neural Network for Sentiment Classification”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Sept. 2015, pp. 1422–1432. DOI: 10.18653/v1/D15-1167.

[54] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler. “Efficient Transformers: A Survey”. In: *ACM Computing Surveys* 55.6 (2022). DOI: 10.1145/3530811.

[55] A. Tripathy, A. Agrawal, and S. K. Rath. “Classification of sentiment reviews using n-gram machine learning approach”. In: *Expert Systems with Applications* 57 (2016), pp. 117–126. DOI: 10.1016/j.eswa.2016.03.028.

[56] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

[57] Y. Yang, W.-t. Yih, and C. Meek. “Wikiqa: A challenge dataset for open-domain question answering”. In: *Proceedings of the 2015 conference on empirical methods in natural language processing*. 2015, pp. 2013–2018.

[58] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. "Xlnet: Generalized autoregressive pretraining for language understanding". In: *Advances in neural information processing systems* 32 (2019).

[59] W. Yin, K. Kann, M. Yu, and H. Schütze. "Comparative study of CNN and RNN for natural language processing". In: *arXiv preprint arXiv:1702.01923* (2017).

[60] T. Young, D. Hazarika, S. Poria, and E. Cambria. "Recent trends in deep learning based natural language processing [Review Article]". In: *IEEE Computational Intelligence Magazine* 13.3 (2018), pp. 55–75. DOI: 10.1109/MCI.2018.2840738.

[61] Y. Yu, X. Si, C. Hu, and J. Zhang. "A review of recurrent neural networks: LSTM cells and network architectures". In: *Neural computation* 31.7 (2019), pp. 1235–1270.