National Technical University of Athens
School of Electrical and Computer Engineering
Department of Signals, Control and Robotics

Unsupervised Domain Adaptation

for

Natural Language Processing

DIPLOMA THESIS

CONSTANTINOS F. KAROUZOS

Supervisor : Alexandros Potamianos
Associate Professor NTUA

Athens, November 2020

National Technical University of Athens
School of Electrical and Computer Engineering
Department of Signals, Control and Robotics

Unsupervised Domain Adaptation for Natural Language Processing

DIPLOMA THESIS

CONSTANTINOS F. KAROUZOS

Supervisor : Alexandros Potamianos
Associate Professor NTUA

Approved by the examining committee on the November 2, 2020.

....	....	....
Alexandros Potamianos Associate Professor NTUA	Constantinos Tzafestas Associate Professor NTUA	Haris Papageorgiou Researcher A', RC Athena

Athens, November 2020

....

Constantinos F. Karouzos

Electrical and Computer Engineer

Dedicated to the memory of my father, Fotis.

Περίληψη

Σκοπός της παρούσας εργασίας είναι η μελέτη του προβλήματος της μη επιβλεπόμενης προσαρμογής πεδίου (unsupervised domain adaptation) για εφαρμογές επεξεργασίας φυσικής γλώσσας και συγκεκριμένα για το πρόβλημα της ανάλυσης συναισθήματος (sentiment analysis). Στο πρόβλημα προσαρμογής πεδίου υπάρχουν δεδομένα που έρχονται από δύο κατανομές, μία κατανομή πηγή (source domain) και μία κατανομή στόχο (target domain), ενώ επισημειώσεις είναι διαθέσιμες μόνο για την κατανομή πηγή. Το πρόβλημα έγκειται στην εκμάθηση, με αξιοποίηση των δεδομένων από τα δύο πεδία, ενός μοντέλου μηχανικής μάθησης με καλή γενίκευση σε δεδομένα που ανήκουν στην κατανομή στόχο. Στην παρούσα διπλωματική εργασία μελετάμε αρχικά το υπόβαθρο μηχανικής μάθησης, σε επίπεδο αρχιτεκτονικών μοντέλων, αλγορίθμων εκπαίδευσης και τεχνικών μάθησης. Στην συνέχεια καλύπτουμε το υπόβαθρο εξελίξεων στο αντικείμενο της επεξεργασίας φυσικής γλώσσας, μέσω μίας αναφοράς σε διανύσματα λέξεων, σε γλωσσικά μοντέλα και τέλος σε προεκπαιδευμένα γλωσσικά μοντέλα και το σύστημα αναπαραστάσεων λέξεων BERT. Για την επίλυση του προβλήματος προσαρμογής πεδίου έχουν προταθεί μια ποικιλία προσεγγίσεων επίλυσης. Αυτές χωρίζονται σε τρεις κύριες κατηγορίες, όσες επιδιώκουν να μάθουν πρώτα τα κοινά χαρακτηριστικά (pivots) μεταξύ των πεδίων, εκείνες που αναπτύσουν μοντέλα ακολουθώντας την λογική της αντιπαραθετικής μηχανικής μάθησης μεταξύ των πεδίων (domain adversarial training) και τέλος στην κατηγορία προσεγγίσεων με βάση τα δεδομένα που επιδιώκουν συνήθως είτε την εκμάθηση ετικετών (pseudo-labels) των παραδειγμάτων της κατανομής στόχου είτε την αξιοποίηση προ-εκπαιδευμένων γλωσσικών μοντέλων.

Στην παρούσα εργασία προτείνουμε μια νέα προσέγγιση για την επίλυση του προβλήματος προσαρμογής πεδίου, βασισμένη στο προεκπαιδευμένο σύστημα αναπαραστάσεων λέξεων BERT. Αυτή αποτελείται από δύο βήματα. Το πρώτο βήμα αφορά την συνέχεια της προεκπαίδευσης μέσω της γλωσσικής μοντελοποίησης των δεδομένων που προέρχονται από την κατανομή στόχο. Το δεύτερο βήμα αποτελείται από την μάθηση της ταξινόμησης από τα δεδομένα από την κατανομή πηγή ενώ συνεχίζεται η γλωσσική μοντελοποίηση στα δεδομένα από την κατανομή στόχο. Το πειραματικό μέρος αυτής της εργασίας περιλαμβάνει ένα σύνολο συγκριτικών πειραμάτων μεταξύ της προτεινόμενης μεθόδου, και ενός συνόλου προηγούμενων μεθόδων ή μεθόδων βάσης. Τα πειράματα αφορούν το multi-domain Amazon reviews dataset που περιέχει κριτικές από πολλές θεματικές ενότητες (βιβλία, ταινίες, ηλεκτρονικά, είδη κουζίνας). Τα αποτελέσματα των παραπάνω πειραμάτων καταδεικνύουν σημαντική βελτίωση των ποσοστών επιτυχίας της προτεινόμενης μεθόδου σε σχέση με τις συγκρινόμενες. Η εργασία περιλαμβάνει ακόμα ανάλυση των αποτελεσμάτων και οπτικοποίηση των χαρακτηριστικών που εξάγονται και χρησιμοποιούνται για ταξινόμηση σε κάθε περίπτωση. Τέλος πραγματοποιούμε μια ανάλυση για τους λόγους αποτυχίας της κυρίαρχης βιβλιογραφικά επιλογής του domain adversarial training βασισμένοι στην σχετική θεωρία μάθησης από διαφορετικά πεδία καθώς και τα πειραματικά μας αποτελέσματα.

Λέξεις κλειδιά

Επεξεργασία Φυσικής Γλώσσας, Μη Επιβλεπόμενη Προσαρμογή Πεδίου, Προσαρμογή Πεδίου, Ανάλυση Συναισθήματος, Μηχανική Μάθηση, Γλωσσικά Μοντέλα, Μη Επιβλεπόμενη Μάθηση, Μάθηση Πολλαπλών Εργασιών

Abstract

The purpose of this diploma dissertation is to study unsupervised domain adaptation for natural language processing applications and specifically for the problem of sentiment analysis. In the domain adaptation problem there is data coming from two distributions, one source domain and one target domain, while labels are only available for the source domain. The aim is learning, by using data from both domains, a model with good generalization on examples belonging to the target domain. In this dissertation we first study the theoretical background of machine learning, at the level of architectural models, training algorithms and learning techniques. Then we cover the background of developments in the subject of natural language processing, making a reference to word vectors, language models and finally to pretrained language models and BERT (Bidirectional Encoder Representations from Transformers). To solve the domain adaptation problem, the literature has proposed a variety of approaches. These are divided into three main categories, those that seek to first learn the common features (pivots) between domains, those that develop models following domain adversarial training and finally the category of data-based approaches which usually seek either to learn pseudo-label for the target domain or the use of pretrained language models.

In the present work we propose a new approach to achieve domain adaptation, based on BERT. It consists of two steps. The first step is the continuation of pretraining through masked language modeling on the data derived from the target domain. On a final fine-tuning step we learn the task on source labeled data, while we keep an auxiliary masked language modeling objective on unlabeled target data. The experimental part of this work includes a set of comparative experiments between the proposed method, and a set of previous methods and baselines. The experiments are conducted on the multi domain (books, movies, electronics, kitchenware) sentiment analysis Amazon reviews dataset. The results of the above experiments show a significant improvement in the accuracy of the proposed method compared to the previous state-of-the-art. The work also includes an analysis of the results and visualization of the features that are extracted and used for classification in each case. Finally, we discuss the limitations of the dominant approach of domain adversarial training, based on the the relevant learning theory from different domains and our experimental observations.

Key words

Natural Language Processing, Domain Adaptation, Unsupervised Domain Adaptation, Sentiment Analysis, Machine Learning, Language Modeling, Unsupervised Learning, Multitask Learning

Ευχαριστίες

Θα ήθελα πρώτα από όλα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή αυτής της διπλωματικής εργασίας, κ. Α. Ποταμιάνο, για την ευκαιρία να πραγματοποιήσω έρευνα στο αντικείμενο της Επεξεργασίας Φυσικής Γλώσσας, τις συμβουλές και την καθοδήγησή του, που υπήρξαν καθοριστικοί παράγοντες προς την επιτυχή ολοκλήρωσή της.

Ακόμα οφείλω να ευχαριστήσω τον υποψήφιο διδάκτορα, συνεργάτη και φίλο Γιώργο Παρασκευόπουλο για την στενή και καρποφόρα συνεργασία, την υπομονή και υποστήριξή του. Θα ήθελα να επισημάνω πως ήταν παρών όχι μόνο με λύσεις και απαντήσεις για μικρά και μεγάλα ζητήματα, αλλά και ως πρότυπο ερευνητή και μηχανικού.

Με αυτή την εργασία κλείνει ο κύκλος των σπουδών μου στην ΣΗΜΜΥ. Κατά την διάρκεια αυτών των χρόνων διασταυρώθηκαν οι δρόμοι μας και συμπορευτήκαμε με αρκετούς ανθρώπους, χωρίς τους οποίους τίποτα δεν θα ήταν ίδιο. Η κοινή ακαδημαϊκή μας πορεία, οι αμέτρητες ώρες εκπόνησης εργασιών και μελέτης, αποτελούν μόνο την αφορμή για τις καλές φιλίες που ανακαλώ στα πρόσωπά τους. Τους ευχαριστώ.

Κλείνοντας την λίστα ευχαριστιών, δεν θα μπορούσα να μην αναφέρω την μητέρα μου, Κατερίνα, για την αμέριστη συμπαράστασή της, καθώς και όσους, ακόμα και χωρίς δεσμούς αίματος, θεωρούνται οικογένεια μου.

Η διπλωματική αυτή εργασία αφιερώνεται στην μνήμη του πατέρα μου, Φώτη.

Κωνσταντίνος Καρούζος,

Αθήνα, 2η Νοεμβρίου 2020

Contents

List of Tables

List of Figures

Εκτεταμένη Ελληνική Περίληψη

0.1 Εισαγωγή

Στην παρούσα εργασία ερευνούμε το πρόβλημα της προσαρμογής γλωσσικού πεδίου χωρίς επιτήρηση (unsupervised domain adaptation) με χρήση προεκπεδευμένων γλωσσικών μοντέλων (pretrained language models) για εφαρμογές επεξεργασίας φυσικής γλώσσας (natural language processing). Προτείνουμε μία νέα μέθοδο προσαρμογής προεκπαιδευμένων γλωσσικών μοντέλων, συνδυάζοντας το κόστος ταξινόμησης και το κόστος γλωσσικής μοντελοποίησης. Η προτεινόμενη μέθοδος επιτυγχάνει προσαρμογή στην κατανομή στόχο με ευρωστία, ακόμα και όταν τα διαθέσιμα δεδομένα είναι λίγα. Επιπρόσθετα παραθέτουμε στοιχεία για για την επίτευξη καλύτερης επαλήθευσης (validation) με χρήση της προτεινόμενης μεικτής συνάρτησης κόστους. Τέλος, πραγματοποιούμε μία ανάλυση για τα όρια της βιβλιογραφικά κυρίαρχης μεθόδου ανταγωνιστικής μηχανικής μάθησης μεταξύ των πεδίων, βασισμένη σε θεωρητικά θεμέλια και πειραματικά αποτελέσματα. Δοκιμάζουμε την προτεινόμενη μέθοδο στα δώδεκα σενάρια προσαρμογής του συνόλου δεδομένων κριτικών από πολλές θεματικές ενότητες multi-domain Amazon Reviews Sentiment datasest. Η μέθοδος πετυχαίνει μέση ακρίβεια 91.73%, το οποίο ισοδυναμεί με 1.10% απόλυτη βελτίωση σε σχέση με την υπάρχουσα βέλτιστη μέθοδο.

Οι βαθιές αρχιτεκτονικές νευρωνικών δικτύων έχουν επιτυχεί βέλτιστα αποτελέσματα σε ένα μεγάλο εύρος εφαρμογών μηχανικής μάθησης. Αν και η μεγάλη πλειοψηφία των μοντέλων βαθιάς μάθησης εκπαιδεύονται και αξιολογούνται σε δεδομένα προερχόμενα από μία συγκεκριμένη κατανομή, τα μοντέλα σε πραγματικές εφαρμογές συχνά χρησιμοποιούνται σε καταστάσεις στις οποίες τα δεδομένα δεν ανήκουν στην κατανομή εκπαίδευσης, γεγονός που οδηγεί σε μείωση της απόδοσής τους. Η συγκέντρωση δεδομένων και η επισημείωση τους, για την εκπαίδευση μοντέλων εξειδικευμένων σε ένα πεδίο, είναι μία διαδικασία υψηλού κόστους που απαιτεί επιπρόσθετο χρόνο και εμποδίζει την επαναχρησιμοποίηση των εκπαιδευμένων μοντέλων, ενώ αντίθετα δεδομένα χωρίς επισημείωση είναι εύκολα προσβάσιμα. Η μη επιβλεπόμενη προσαρμογή γλωσσικού πεδίου είναι συνεπώς ένας ενεργός τομέας έρευνας με υψηλό αντίκτυπο στην πραγματική προσαρμογή των μοντέλων μηχανικής μάθησης.

Συγκεκριμένα η εργασία αυτή αφορά την μη επιβλεπόμενη προσαρμογή πεδίου δικτύων βασισμένων σε προεκπαιδευμένα γλωσσικά μοντέλα. Τα προεκπαιδευμένα γλωσσικά μοντέλα [24, 39, 98, 56, 19] αποτελούν μια επαναστατική τομή στην εξέλιξη της έρευνας για την επεξεργασία φυσικής γλώσσας. Προεκπαιδεύονται σε μεγάλα σύνολα κειμενικών δεδομένων, κωδικοποιούν πληροφορίες σχετικές με τα χαρακτηριστικά της γλώσσας, τα συμφραζόμενα, την σύνταξη και την σημασιολογία, και όταν προσαρμόζονται μέσω μεταφοράς μάθησης, επιτρέπουν σημαντικές βελτιώσεις σε πολλές εργασίες επεξεργασίας γλώσσας μέσω της βελτιστοποίησης τους με λίγα δεδομένα. Αυτή η διαδικασία βελτιστοποίησης εισήγαγε την ανάγκη για διαδοχική μεταφορά μάθησης, η οποία προσαρμόζει τα προεκπαιδευμένα μοντέλα σε νέα πεδία και τελικές εργασίες στόχους. Η βελτιστοποίηση των γλωσσικών μοντέλων παρέχει συνεπώς ένα απλό και άμεσο πλαίσιο για ημι-επιβλεπόμενη προσαρμογή πεδίου, κατά την οποία παρότι μαθαίνουμε νέες εργασίες από επισημειωμένα δεδομένα, εκμεταλλευόμαστε και την δυναμική των γλωσσικών μοντέλων ως μη επιβλεπόμενα συστήματα πολλαπλών εργασιών. Ωστόσο, παρά την κωδικοποιημένη γνώση από πολλές θεματικές που περιέχεται σε ένα σύγχρονο προεκπαιδευμένο γλωσσικό μοντέλο, η μάθηση εργασιών από δεδομένα εκτός πεδίου παραμένει ένα απαιτητικό ζήτημα.

Προτείνουμε μία απλή και αποτελεσματική προσέγγιση για την μη επιβλεπόμενη προσαρμογή πεδίου, που αξιοποιεί τις δυνατότητες των υψηλής χωρητικότητας προεκπαιδευμένων γλωσσικών μοντέλων τα οποία βασίζονται στην αρχιτεκτονική Transformer [92], όπως το BERT [24]. Ξεκινώντας από ένα προεκπαιδευμένο σε γενικά κειμενικά δεδομένα μοντέλο BERT, προτείνουμε την συνέχεια της διαδικασίας προεκπαίδευσης μέσω της γλωσσικής μοντελοποίησης με μάσκες σε λίγα διαθέσιμα σχετικά δεδομένα χωρίς επισημείωση του πεδίου στόχου. Κατά την διάρκεια της τελικής βελτιστοποίησης, ενώ μαθαίνουμε την εργασία από τα δεδομένα πηγής με επισημείωση, διατηρούμε ως βοηθητική την εργασία γλωσσικής μοντελοποίησης με μάσκες στα δεδομένα του πεδίου στόχου.

Οι σημαντικότερες συνεισφορές είναι: (1) προτείνουμε μία νέα, απλή και εύρωστη μέθοδο μη επιβλεπόμενη προσαρμογής πεδίου με χρήση γλωσσικής μοντελοποίησης, (2) επιτυγχάνουμε βέλτιστα αποτελέσματα στο multi-domain Amazon reviews dataset, ξεπερνώντας σε απόδοση πιο περίπλοκες προσεγγίσεις και (3) επιχειρηματολογούμε για τα όρια της ανταγωνιστικής μηχανικής μάθησης μεταξύ πεδίων, βασισμένοι στην θεωρία μάθησης από διαφορετικά πεδία και τις πειραματικές παρατηρήσεις μας.

0.2 Σχετική βιβλιογραφία

Οι Ben-David κ.ά. [12, 10] παρέχουν μία θεωρία μάθησης από διαφορετικά πεδία. Οι Ganin κ.ά. [30, 29] προτείνουν την εκμάθηση εργασιών, ενώ το δίκτυο δεν είναι σε θέση να διακρίνει αν τα παραδείγματα ανήκουν στο πεδίο πηγή ή στο πεδίο στόχο, με χρήση ενός επιπρόσθετου ανταγωνιστικού κόστους. Σύμφωνα με τους Ramponi και Plank [72] η ανταγωνιστική μάθηση μεταξύ πεδίων έχε κυρίαρχη θέση στην βιβλιογραφία της μη επιβλεπόμενης προσαρμογής πεδίου για γλωσσικές εφαρμογές [48, 5, 82]. Πρόσφατα, οι Du κ.ά. [26] πρότειναν την χρήση ανταγωνιστικής μάθησης μεταξύ πεδίων στο πλαίσιο ενός μοντέλου BERT. Οι Zhao κ.ά. [102] προτείνουν ανταγωνιστικής μάθηση μεταξύ πεδίων από πολλά πεδία πηγές. Οι Guo κ.ά. [35] χρησιμοποιούν μία προσέγγιση μίξης ειδικών μοντέλων για προσαρμογή πεδίου από πολλά πεδία πηγές. Οι Guo κ.ά. [34] διερευνούν μέτρα απόστασης ως πρόσθετα κόστη καθώς και την χρήση ενός δυναμικού ελεγκτή πεδίων. Οι Shen κ.ά. [86] μαθαίνουν σταθερά χαρακτηριστικά ανεξαρτήτως πεδίου με χρήση της απόστασης Wasserstein. Οι Bousmalis κ.ά. [18] εισάγουν δίκτυα διαχωρισμού πεδίου με ιδιωτικούς και κοινόχρηστους κωδικοποιητές.

Μία άλλη γραμμή έρευνας για την μη επιβλεπόμενη προσαρμογή πεδίου, επικεντρώνεται στην εξαγωγή κοινών χαρακτηριστικών μέσω της εκμάθησης σταθερών στοιχείων (pivots). Το Structural Correspondence Learning (SCL) [15] και το Spectral Feature Alignment [65] είναι μεταξύ των αρχικών σχετικών μεθόδων. Οι Ziser και Reichart [105, 106, 108] συνδυάζουν το SCL με αρχιτεκτονικές νευρωνικών δικτύων και την λογική της γλωσσικής μοντελοποίησης. Ο Miller [62] προτείνει την από κοινού εκμάθηση της βασικής εργασίας και των pivot χαρακτηριστικών. Οι Li κ.ά. [53] μαθαίνουν τα pivot χαρακτηριστικά με ιεραρχικά δίκτυα προσοχής. Οι μέθοδοι με pivot χαρακτηριστικά έχουν επίσης χρησιμοποιηθεί σε συνδυασμό με το BERT στο Ben-David κ.ά. [9].

Οι τεχνικές ψευδών επισημειώσεων (pseudo-labeling) είναι αλγόριθμοι ημι-επιβλεπόμενης μάθησης που χρησιμοποιούν το ίδιο μοντέλο στην περίπτωση της αυτοεκπαίδευσης [99, 58, 2] είτε πολλαπλά αρχικά μοντέλα στην περίπτωση της τριπλής εκπαίδευσης [103, 89] ως οδηγό για την επισημείωση των δεδομένων του πεδίου στόχου. Οι Saito κ.ά. [81] προτείνουν μια ασύμμετρη προσέγγιση τριπλής εκπαίδευσης. Οι Ruder και Plank [79] εισάγουν μία προσέγγιση τριπλής εκπαίδευσης με λογική εκπαίδευσης πολλαπλών εργασιών. Οι Rotman και Reichart [76] καθώς και οι Lim κ.ά. [55] μελετούν τεχνικές ψευδών επισημειώσεων σε συνδυασμό με αναπαραστάσεις λέξεων από τα συμφραζόμενα. Οι Ye κ.ά. [101] συνδυάζουν την τεχνική ψευδών επισημειώσεων με αυτοεκπαίδευση με προεκπαιδευμένα γλωσσικά μοντέλα (XLM-R [23]) και προτείνουν το CFd.

Η προεκπαίδευση και η τελική βελτιστοποίηση είναι ένας απλός και άμεσος τρόπος για προσαρμογή. Όταν αρκετά δεδομένα είναι διαθέσιμα σε ένα γλωσσικό πεδίο, η προεκπαίδευση ενός μοντέλου από την αρχή είναι μια απλή και αποτελεσματική ιδέα. Μοντέλα για συγκεκριμένα πεδία, βασισμένα στο BERT, όπως το BioBERT [45] και το SciBERT [8], το έχουν δοκιμάσει με επιτυχία. Οι Sun κ.ά. [91] προτείνουν την συνέχιση της προεκπαίδευσης του BERT με δεδομένα από το πεδίο στόχο και την χρήση επιπλέον σχετικών εργασιών. Οι Xu κ.ά. [97] εισάγουν μία βοηθητική εργασία κατανόησης κριτικών και προτείνουν την συνέχιση της προεκπαίδευσης του BERT με ένα πρόσθετο κόστος σε μία εργασία ερωτήσεων και απαντήσεων. Η συνέχεια της προεκπαίδευσης σε πολλές φάσεις, από γενικά σε συγκεκριμένα δεδομένα και δεδομένα συγκεκριμένα για την εργασία στόχο, επιπλέον βοηθάει στην απόδοση των προεκπαιδευμένων γλωσσικών μοντέλων, σύμφωνα με τους Gururangan κ.ά. [36]. Το AdaptaBERT [38] προτείνει ένα δεύτερο βήμα προεκπαίδευσης του BERT για να επιτύχει μη επιβλεπόμενη προσαρμογή πεδίου. Το Multi-Task Learning (MTL) [20] μέσω της κοινής χρήσης παραμέτρων σε νευρωνικά δίκτυα έχει αποδειχθεί εξαιρετικής αποτελεσματικότητας [77]. Η γλωσσική μοντελοποίηση ως βοηθητική εργασία προτείνεται για να αποφευχθεί το catastrophic forgetting κατά την μεταφορά μάθησης [22]. Επίσης έχει χρησιμοποιηθεί [40] για την επίλυση του προβλήματος αναγνώρισης οντοτήτων από διαφορετικά γλωσσικά πεδία.

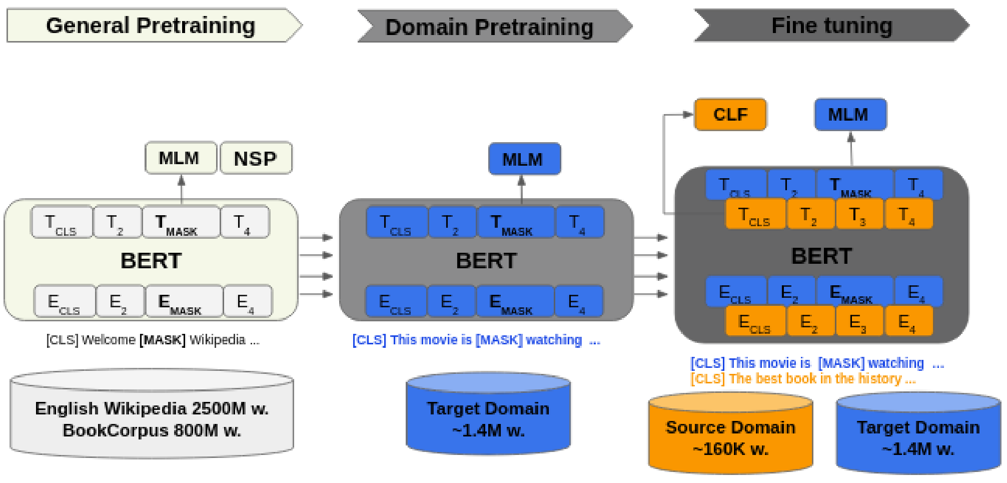
0.3 Ορισμός Προβλήματος

Έστω X ο χώρος εισόδων και Y το σύνολο των επισημειώσεων, για δυαδική ταξινόμηση $Y = \{0, 1\}$. Στην προσαρμογή πεδίου υπάρχουν δύο διαφορετικές κατανομές στο $X \times Y$, που ονομάζονται πεδίο πηγή source domain D_S και πεδίο στόχος target domain D_T. Στο σενάριο μη επιβλεπόμενης μάθησης παρέχονται επισημειώσεις για δείγματα που προέρχονται από το D_S, ενώ τα δείγματα που προέρχονται από το D_T είναι χωρίς επισημείωση. Ο στόχος είναι να εκπαιδευτεί ένα μοντέλο που έχει καλή απόδοση σε δείγματα που προέρχονται από το πεδίο στόχο D_T. Αυτό συνοψίζεται στις ακόλουθες εξισώσεις:

$$\begin{aligned} S &= (x_i, y_i)_{i=1}^{n} \sim (D_S)^n \\ T &= (x_i)_{i=n+1}^{n+m} \sim (D_T^X)^m \end{aligned} \tag{0.1}$$

όπου D_T^X είναι η οριακή κατανομή του D_T στο X, n είναι ο αριθμός δειγμάτων από το πεδίο πηγή και m ο αριθμός των δειγμάτων από το πεδίο στόχο.

0.4 Προτεινόμενη μέθοδος


General Pretraining
Domain Pretraining
Fine tuning
MLM
NSP
MLM
CLF
MLM
BERT
BERT
BERT
[CLS] Welcome [MASK] Wikipedia ...
[CLS] This movie is [MASK] watching ...
[CLS] This movie is [MASK] watching ...
[CLS] The best book in the history ...
English Wikipedia 2500M w.
BookCorpus 800M w.
Target Domain ~1.4M w.
Source Domain ~160K w.
Target Domain ~1.4M w.

Σχήμα 0.1: a) Το BERT [24] προεκπαιδεύεται στην αγγλική Wikipedia και το BookCorpus με τις εργασίες MLM και NSP. b)Συνεχίζουμε την προεκπαίδευση του BERT σε δεδομένα από την κατανομή στόχο με την εργασία MLM. c) Εκπαιδεύουμε έναν ταξινομητή με τα δεδομένα από το πεδίο πηγή, ενώ διατηρούμε την εργασία MLM στα δεδομένα χωρίς επισημειώσεις του πεδίου στόχου.

Το Σχήμα 0.1. δίνει μια επισκόπηση της προσέγγισής μας για την μη επιβλεπόμενη προσαρμογή πεδίου, που αποτελείται από δύο βήματα. Ξεκινώντας από ένα προεκπαιδευμένο μοντέλο, συνεχίζουμε να το εκπαιδεύουμε σε δεδομένα του πεδίου στόχου με την εργασία γλωσσικής μοντελοποίησης με μάσκες. Σε ένα δεύτερο και τελευταίο βήμα βελτιστοποίησης μαθαίνουμε την εργασία ταξινόμησης σε δεδομένα με επισημείωση του πεδίου πηγής, ενώ διατηρούμε ως βοηθητική την εργασία γλωσσικής μοντελοποίησης με μάσκες στα δεδομένα χωρίς επισημείωση του πεδίου στόχου.

Το BERT είναι προεκπαιδευμένο με χρήση των εργασιών γλωσσικής μοντελοποίησης με μάσκες (MLM task) και ταξινόμησης ανάλογα με το αν ένα ζευγάρι προτάσεων εισόδου είναι συνεχόμενες ή όχι (NSP task). Ένα προεκπαιδευμένο μοντέλο BERT μπορεί να βελτιστοποιηθεί με ένα μόνο επίπεδο εξόδου για την επίλυση ενός ευρέος φάσματος εργασιών. Λαμβάνουμε ένα προεκπαιδευμένο μοντέλο BERT ως αρχικοποίηση και πραγματοποιούμε ένα δεύτερο βήμα προεπεξεργασίας για το πεδίο στόχο. Αυτό το βήμα γίνεται με την εργασία MLM σε δεδομένα χωρίς επισημείωση του πεδίου στόχου. Σε αντίθεση με το AdaptaBERT [38] και το BERT-DAAT [26] χρησιμοποιούμε δεδομένα μόνο από το πεδίο στόχο κατά την διάρκεια αυτού του βήματος.

Στη συνέχεια, μεταφέρουμε τα βάρη του επιπρόσθετα προεκπαιδευμένου μοντέλου BERT και προσθέτουμε έναν τελικό ταξινομητή. Ακολουθώντας την τυπική πρακτική για BERT, περνάμε την αναπαράσταση του τελικού επιπέδου $[CLS]$ στον ταξινομητή για την συγκεκριμένη εργασία ταξινόμησης. Ο ταξινομητής αποτελείται από ένα γραμμικό στρώμα τροφοδοσίας προς τα εμπρός με χρήση dropout. Διατηρούμε επίσης το τελικό στρώμα του BERT για την εργασία γλωσσικής μοντελοποίησης με μάσκες.

Προκειμένου να επιτευχθεί η προσαρμογή στο πεδίο στόχο και να βελτιωθεί η ικανότητα γενίκευσης του μοντέλου, διατηρούμε τον στόχο γλωσσικής μοντελοποίησης με μάσκες στο πεδίο στόχο, ενώ μαθαίνουμε την εργασία στο πεδίο πηγή. Το μοντέλο εκπαιδεύεται από τα δεδομένα με επισημείωση στο πεδίο πηγή για την εργασία ταξινόμησης και τα δεδομένα χωρίς επισημείωση στο πεδίο στόχο για την εργασία γλωσσικής μοντελοποίησης με μάσκες. Χρησιμοποιούμε μάσκες μόνο για τα δεδομένα από το πεδίο στόχος. Κατά τη διάρκεια της εκπαίδευσης, παρεμβάλλουμε δεδομένα πηγής και στόχου, και τα δύο διέρχονται μέσω του BERT. Τα δεδομένα πηγής με επισημείωση προωθούνται στον ταξινομητή, ενώ τα δεδομένα στόχου χωρίς επισημείωση προωθούνται στην έξοδο MLM.

Η κοινή συνάρτηση κόστους είναι το άθροισμα της συνάρτησης κόστους ταξινόμησης L_{CLF} και της συνάρτησης κόστους γλωσσικής μοντελοποίησης MLM L_{MLM}. Εκπαιδεύουμε το μοντέλο σε μικτές παρτίδες batch, που περιλαμβάνουν δεδομένα πηγής και στόχου, που χρησιμοποιούνται για τις αντίστοιχες εργασίες. Διαιρούμε ένα batch σε πολλα δευτερεύοντα και συγκεντρώνουμε τις παραγώγους αυτών των sub-batches πριν από τις ενημερώσεις των παραμέτρων. Δίνουμε m sub-batches πηγής με επισημείωση και n sub-batches στόχου χωρίς επισημείωση σε ένα batch. Ως εκ τούτου, η κοινή συνάρτηση κόστους με την οποία εκπαιδεύεται το μοντέλο είναι:

$$L(\mathbf{s}, \mathbf{t}) = \lambda L_{CLF}(\mathbf{s}) + (1 - \lambda) L_{MLM}(\mathbf{t}) \tag{0.2}$$

όπου ο συντελεστής στάθμισης λ υπολογίζεται με βάση τον αριθμό source και target δειγμάτων.

$$\lambda = \frac{n}{n+m} \tag{0.3}$$

0.5 Πειράματα

0.5.1 Σύνολο Δεδομένων

Αξιολογούμε την μέθοδο στο multi-domain Amazon reviews dataset [14], ένα από τα χαρακτηριστικά σύνολα δεδομένων αναφοράς για προσαρμογή πεδίου. Οι κριτικές με 1 ή 2 αστέρια κατατάσσονται ως αρνητικές, ενώ οι κριτικές με 4 ή 5 αστέρια κατατάσσονται ως θετικές. Το σύνολο δεδομένων περιέχει κριτικές σε τέσσερα πεδία - θεματικές ενότητες : *Books* (B), *DVDs* (D), *Electronics* (E) και *Kitchen appliances* (K), δίνοντας 12 σενάρια προσαρμογής. Χρησιμοποιούμε και τις 2.000 κριτικές με επισημείωση (1.000 θετικές και 1.000 αρνητικές) και 19809 B, 19798 D, 19937 E, 17805 K, τυχαία επιλεγμένες κριτικές χωρίς επισημείωση. Τα δεδομένα με επισημείωση χρησιμοποιούνται για τεστ, στα σενάρια στα οποία το αντίστοιχο πεδίο θεωρείται στόχος.

0.5.2 Λεπτομέρειες υλοποίησης

Χρησιμοποιούμε το $BERT_{BASE}$ (uncased) ως βάση για να ξεκινήσουμε την προεκπαίδευση στο γλωσσικό πεδίο στόχο. Το αρχικό αγγλικό μοντέλο $BERT_{BASE}$ είναι μια αρχιτεκτονική νευρωνικού δικτύου 12 επιπέδων, 768 κρυφών επιπέδων, 12 κεφαλών, 110 εκατομμυρίων παραμέτρων, εκπαιδευμένη στο BooksCorpus με 800 εκατομμύρια λέξεις και μια έκδοση της αγγλικής Wikipedia με 2.500 εκατομμύρια λέξεις . Ακολουθούμε την αρχικά προτεινόμενη διαδικασία απόκρυψης λέξεων με μάσκες, δηλαδή την τυχαία κάλυψη 15% των token WordPiece [96]. Εάν ένα token σε μια συγκεκριμένη θέση έχει επιλεγεί για να κρυφτεί, 80% των περιπτώσεων αντικαθίσταται με ένα token $[MASK]$, 10% των περιπτώσεων με ένα τυχαίο token και κατά 10% παραμένει αμετάβλητο. Το μέγιστο μήκος ακολουθίας ορίζεται σε 512 με περικοπή των εισόδων. Κατά τη διάρκεια εκτεταμένης προεκπαίδευσης εκπαιδεύουμε με μέγεθος batch 8 για 3 εποχές. Κατά τη διάρκεια του τελικού βήματος εκπαιδεύουμε με μέγεθος batch 36, αποτελούμενο από 4 δείγματα πηγής και 8 sub-batches στόχου 4 δειγμάτων το καθενα. Ενημερώνουμε τις παραμέτρους μετά από κάθε 5 συσσωρευμένα batch.

Για το πείραμα ανταγωνιστικής μάθησης μεταξύ πεδίων, ορίσαμε τον συντελεστή βάρους σε 0.01. Επίσης πειραματιστήκαμε με $\lambda = 1$, $\lambda = 0.1$ και ένα σχέδιο σιγμοειδούς αύξησης του λ. Αναφέρουμε τα καλύτερα αποτελέσματα.

Για την ανάπτυξη των πειραμάτων χρησιμοποιούμε PyTorch [66] και HuggingFace Transformers [95].

0.5.3 Συγκρινόμενες Μέθοδοι

Θα συγκρίνουμε τη μέθοδο μας, με προηγούμενες εργασίες που αντιπροσωπεύουν μεθόδους με την καλύτερη απόδοση από διαφορετικές προσεγγίσεις για την μη επιβλεπόμενη προσαρμογή πεδίου. Συγκεκριμένα συγκρίνουμε τα αποτελέσματα με το βασισμένο σε BERT **R-PERL** [9], το βασισμένο σε BERT με ανταγωνιστική μάθηση πεδίων **BERT-DAAT** [26] και το βασισμένο σε XLM-R [23] **p+CFd** [101]. Επιπλέον υλοποιήσαμε και αναφέρουμε αποτελέσματα για τα ακόλουθα πειράματα:

- **BERT Source Only**: Βελτιστοποιούμε το $BERT_{BASE}$ μόνο με δεδομένα από το πεδίο πηγή (source).
- **DPT BERT**: Ακολουθούμε το πρώτο βήμα της προτεινόμενης μεθόδου, την συνέχεια προεκπαίδευσης με MLM στο πεδίο στόχο. Μετά βελτιστοποιούμε το μοντέλο μόνο με δεδομένα από το πεδίο πηγή.
- **DAT BERT**: Ανταγωνιστική εκπαίδευση του BERT μεταξύ των πεδίων. Ξεκινάμε από το προεκπαιδευμένο στο πεδίο BERT και το βελτιστοποιούμε ακολουθώντας το [29].
- **Προτεινόμενο**: Η βασισμένη σε BERT πρόταση που συνδυάζει την προεκπαίδευση στο πεδίο με την βελτιστοποίηση με χρήση παράλληλης γλωσσικής μοντελοποίησης με μάσκες, όπως αναλύεται στην ενότητα 0.4.

0.6 Αποτελέσματα και Συζήτηση

0.6.1 Σύγκριση με το state-of-the-art

Παρουσιάζουμε αποτελέσματα για τα 12 σενάρια προσαρμογής πεδίου στον Πίνακα 0.1. Η τελευταία γραμμή του Πίνακα 0.1 περιέχει τη μέση όρο ακρίβειας από όλα τα σενάρια προσαρμογής. Το προτεινόμενο μοντέλο, το οποίο συνδυάζει εκτεταμένη προεκπαίδευση στο γλωσσικό πεδίο στόχο και

	R-PERL	DAAT	p+CFd	Source Only	DPT	DAT	Proposed
$B \to D$	87.8%	90.9%	87.7%	90.5%	90.7%	90.7%	**91.3%**
$B \to E$	87.2%	88.9%	**91.3%**	**91.3%**	90.9%	91.1%	91.2%
$B \to K$	90.2%	88.0%	92.5%	91.6%	92.3%	92.8%	**92.9%**
$D \to B$	85.6%	89.7%	**91.5%**	90.2%	90.5%	90.6%	91.4%
$D \to E$	89.3%	90.1%	91.6%	88.5%	91.7%	88.8%	**92.9%**
$D \to K$	90.4%	88.8%	92.5%	90.5%	92.0%	92.0%	**94.3%**
$E \to B$	90.2%	89.6%	88.7%	87.8%	88.3%	89.4%	**90.6%**
$E \to D$	84.8%	**89.3%**	88.2%	87.2%	87.3%	86.5%	88.4%
$E \to K$	91.2%	91.7%	93.6%	92.8%	94.1%	94.6%	**94.8%**
$K \to B$	83.0%	**90.8%**	89.8%	88.6%	89.4%	83.6%	89.4%
$K \to D$	85.6%	**90.5%**	87.8%	87.1%	88.0%	83.6%	89.2%
$K \to E$	91.2%	93.2%	92.6%	91.9%	93.1%	92.4%	**94.3%**
Average	87.50%	90.12%	90.63%	89.83%	90.69%	89.68%	**91.73%**

Πίνακας 0.1: Ακρίβεια του domain adaptation στα δώδεκα ζεύγη του Amazon Reviews Multi Domain Sentiment Dataset.

βελτιστοποίηση με βοηθητικό MLM ξεπερνά όλες τις άλλες τεχνικές, αποδίδοντας μια απόλυτη βελτίωση $1,90\%$ σε σχέση με το απλό μοντέλο BERT. Συγκρίνουμε αποτελέσματα μόνο από μεθόδους που βασίζονται σε προεκπαιδευμένα μοντέλα, κυρίως BERT. Παρατηρούμε ότι το BERT, όταν βελτιστοποιείται μόνο με τα δεδομένα που φέρουν επισημείωση από το πεδίο πηγή χωρίς καμία γνώση του πεδίου στόχου είναι αρκετά ανταγωνιστικό και με την κατάλληλη εκπαίδευση ξεπερνά το R-PERL [9].

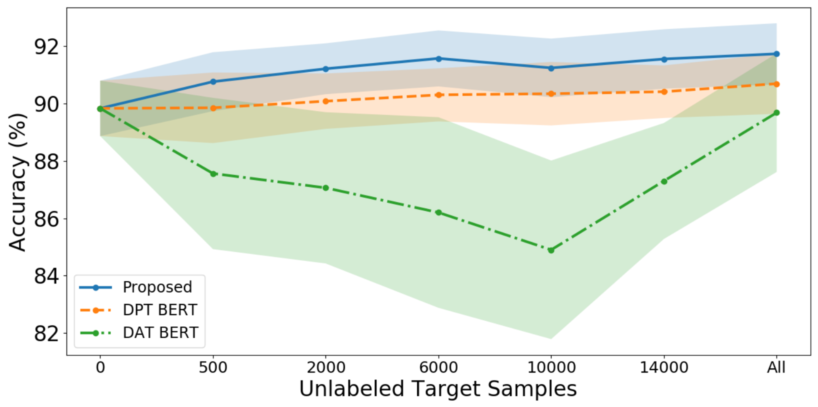
Το πείραμα αντιπαράθεσης πεδίων (Adv.), το οποίο πραγματοποιήθηκε μετά από εκτεταμένο tuning των παραμέτρων του συντελεστή στάθμισης, των βημάτων εκπαίδευσης και του αριθμού των δεδομένων από το πεδίο στόχο που χρησιμοποιήθηκαν, αποδίδει χειρότερα αποτελέσματα από το απλό BERT. Δηλαδή, ακόμη και με προσεκτική επιλογή των υπερπαραμέτρων, η εκπαίδευση του BERT με adversarial training είναι ασταθής και μπορεί να βλάψει τη συνολική απόδοση.

Η συνέχεια της προεκπαίδευσης στο πεδίο στόχο αυξάνει τη μέση ακρίβεια με μία βελτίωση $0,86\%$. Η συνέχεια της προεκπαίδευσης στα δεδομένα του πεδίου στόχου προσαρμόζει καλύτερα το μοντέλο και οδηγεί σε βελτιωμένη απόδοση. Αυτό συμβαδίζει με παρόμοιες προηγούμενες εργασίες για supervised [36, 97, 91] και unsupervised settings [38, 26].

Στην προτεινόμενη μέθοδο, η διατήρηση του MLM κατά το τελικό βήμα βελτιώνει περαιτέρω την απόδοση και επιτυγχάνει μια πρόσθετη αύξηση της μέσης ακρίβειας 1.04%. Η προτεινόμενη μέθοδος, ξεπερνά την ακρίβεια μέσου όρου όλων των άλλων προτεινόμενων προσεγγίσεων για προσαρμογή πεδίου χωρίς επιτήρηση. Συγκεκριμένα, η προτεινόμενη μέθοδος είναι κατά 1.10% καλύτερη από το p+CFd [101], κατά 1.61% από το BERT-DAAT [26] και κατά 4.23% του R-PERL [9].

0.6.2 Αποτελεσματικότητα ως προς τον αριθμό δειγμάτων

Εξετάζουμε περαιτέρω τον αντίκτυπο της χρήσης διαφορετικού αριθμού δεδομένων χωρίς επισημείωση του πεδίου στόχου στην απόδοση των εξεταζόμενων μεθόδων. Πειραματιστήκαμε με 500, 2000, 6000, 10000 και 14000 δείγματα, περιορίζοντας τυχαία τον αριθμό των δεδομένων χωρίς επισημείωση του πεδίου στόχου. Για κάθε μέγεθος dataset πραγματοποιήσαμε τρία πειράματα με μοντέλα BERT: (1) εκτεταμένη προεκπαίδευση πεδίου (Domain Pretraining), (2) domain adversarial training και (3) την προτεινόμενη μέθοδο. Και πάλι, όπως στην κύρια πειραματική ενότητα, δεν ρυθμίσαμε τις υπερ-παραμέτρους για την προεκπαίδευση στο πεδίο στόχο ή την προτεινόμενη μέθοδο. Το Σχήμα 0.3 δείχνει τη μέση ακρίβεια στα δώδεκα σενάρια προσαρμογής του συνόλου δεδομένων που μελετήθηκε. Βλέπουμε ότι η προτεινόμενη μέθοδος είναι ισχυρή ακόμη και σε σενάρια λίγων δεδομένων, καθώς αποδίδει αντίστοιχη βελτίωση ακόμη και στο πείραμα με 500 δείγματα.



Σχήμα 0.2: Μέση ακρίβεια των μεθόδων ανάλογα με τον αριθμό δειγμάτων από το πεδίο στόχο.

0.6.3 Validation

Ένα κοινό πρόβλημα για την μη επιβλεπόμενη προσαρμογή πεδίου είναι η έλλειψη επισημειωμένων δεδομένων που να μπορούν να χρησιμοποιηθούν ως σύνολο επικύρωσης (validation set) προκειμένου να καθοριστεί εάν η συνέχιση της διαδικασίας εκπαίδευσης ωφελεί ή βλάπτει τη βελτιστοποίηση της απόδοσης των μοντέλων. Επιπλέον, κατά την εκπαίδευση υπό αλλαγή πεδίου, η βελτιστοποίηση της απόδοσης στο πεδίο πηγή ενδέχεται να μην έχει βέλτιστη απόδοση για δείγματα που λαμβάνονται από το πεδίο στόχο.

Stopping Criterion	Epochs	Av. Acc.
Fixed	1	90.98
Fixed	3	91.65
Fixed	10	**91.75**
Min source loss	10, patience 3	91.30
Min mixed loss	10, patience 3	**91.73**

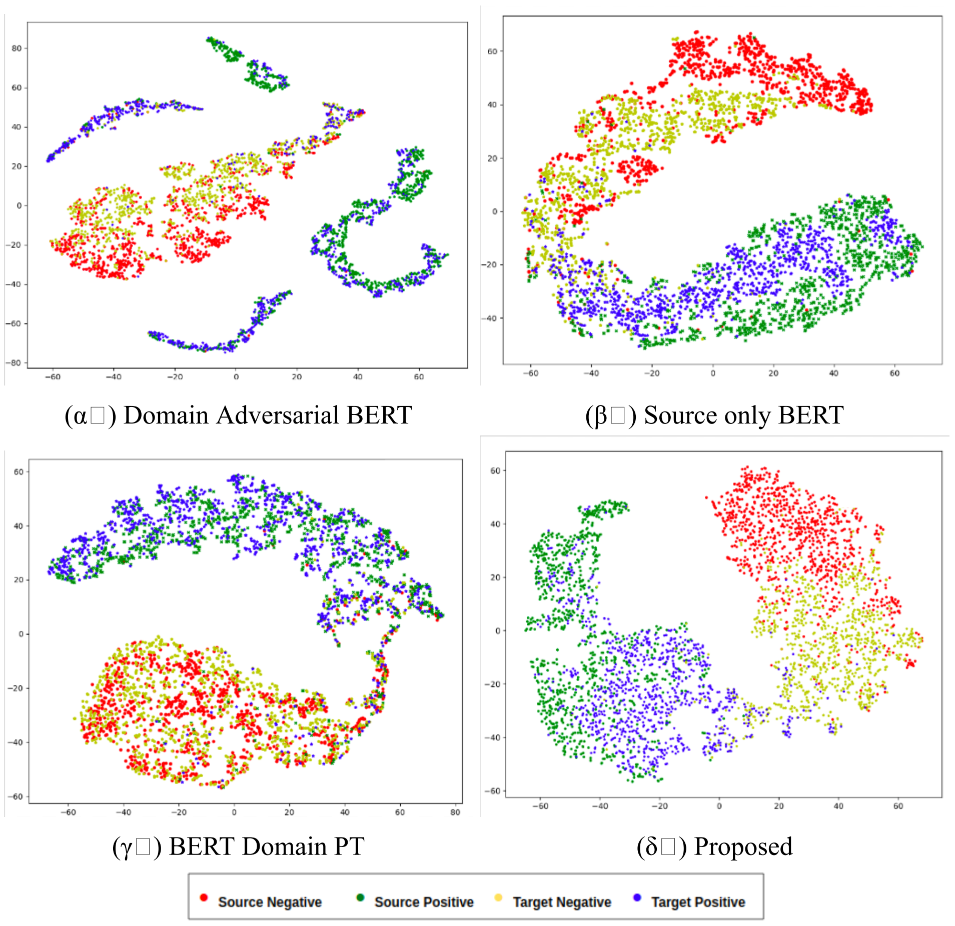
Πίνακας 0.2: Σύγκριση της μέσης ακρίβειας για διάφορες μεθόδους validation.

Εξετάζουμε εάν μπορεί να πραγματοποιηθεί επιτυχώς validation στο προτεινόμενο μικτό loss. Συγκρίνουμε πέντε κριτήρια διακοπής: (1) χωρίς επικύρωση, εκπαίδευση για 1 εποχή, (2) χωρίς επικύρωση, εκπαίδευση για 3 εποχές, (3) επικύρωση στα δεδομένα πεδίου πηγής, εκπαίδευση για 10 εποχές με early stopping και patience 3, (4) επικύρωση σε μικτά δεδομένα, χρησιμοποιώντας το μικτό loss εκπαίδευσης, εκπαίδευση για 10 εποχές με early stopping και patience 3 και (5) χωρίς επικύρωση, εκπαίδευση για 10 εποχές. Αναφέρουμε τα μέσα αποτελέσματα των πέντε μεθόδων στα δώδεκα σενάρια προσαρμογής του συνόλου δεδομένων που μελετήθηκαν στον Πίνακα 0.2. Η επικύρωση δεδομένων τόσο από τα πεδίο πηγή όσο και από το πεδίο στόχο για το μεικτό loss αποφέρει καλύτερα αποτελέσματα κατά μέσο όρο. Έτσι, η ελαχιστοποίηση του προτεινόμενου μικτού loss μπορεί να χρησιμεύσει ως αποτελεσματικό κριτήριο διακοπής κατά την μη επιβλεπόμενη προσαρμογή πεδίου.

0.6.4 Οπτικοποίηση χαρακτηριστικών

Παρουσιάζουμε αναπαραστάσεις t-SNE των χαρακτηριστικών που το BERT έμαθε με διαφορετικές μεθόδους στην σενάριο προσαρμογής $D \rightarrow K$ στο Σχήμα 0.3. Αυτές είναι οι αναπαραστάσεις των χαρακτηριστικών από το token $[CLS]$ των δεδομένων πεδίου πηγής με επισημείωση και των test δεδομένων του πεδίου στόχου.

Στο Σχήμα 0.3.a. φαίνονται οι αναπαραστάσεις που δημιουργούνται κατά το domain adversarial training. Στο Σχήμα 0.3.b παρουσιάζουμε τις αναπαραστάσεις που δημιουργήθηκαν από το BERT όταν εκπαιδεύτηκε μόνο σε δεδομένα από το πεδίο πηγή. Όπως αναμενόταν, τα θετικά και αρνητικά



(α) Domain Adversarial BERT (β) Source only BERT

(γ) BERT Domain PT (δ) Proposed

Σχήμα 0.3: t-SNE visualization των αναπαραστάσεων του BERT $[CLS]$ token για το σενάριο $D \rightarrow K$.

δείγματα του πεδίου πηγή διαχωρίζονται καλά. Αντιθέτως, θετικά και αρνητικά δείγματα από το πεδίο στόχο, ενώ ακολουθούν γενικά τις αντίστοιχες συστάδες των δεδομένων πηγής, επιπλέον μοιράζονται μια κοινή περιοχή. Στο Σχήμα 0.3.c παρουσιάζουμε τις αναπαραστάσεις που δημιουργήθηκαν από το BERT, με συνέχεια της προεκπαίδευσης στο πεδίο στόχο. Αυτή η δεύτερη οπτικοποίηση ακολουθεί το προηγούμενο μοτίβο, αν και παρατηρούμε ότι η διακριτή περιοχή των θετικών και αρνητικών σημείων του πεδίου στόχου μειώνεται σημαντικά.

Στο Σχήμα 0.3.d είναι οι αναπαραστάσεις χαρακτηριστικών που δημιουργήθηκαν από την προτεινόμενη μέθοδο. Τα μπλε και κίτρινα σημεία δεδομένων διαχωρίζονται καλύτερα από ότι στις προηγούμενες περιπτώσεις και η περιοχή ανάμιξης είναι μικρότερη. Έτσι, τα θετικά και αρνητικά δείγματα από το πεδίο στόχο διακρίνονται καλά και αποτελούν μέρος των αντίστοιχων ομάδων με τα σημεία από το πεδίο πηγή. Συνεπώς, το προτεινόμενο μοντέλο προσαρμόστηκε με καλύτερο τρόπο στο πεδίο στόχο και κατάφερε να διαχωρίσει καλύτερα τα θετικά από τα αρνητικά δείγματα.

0.7 Σχετικά με τους περιορισμούς του Domain Adversarial Training

0.7.1 Θεωρία Μάθησης από διαφορετικά πεδία

Οι Ben-David κ.ά. [12, 10] παρουσιάζουν μια θεωρία μάθησης από διαφορετικά πεδία. Ένα βασικό αποτέλεσμα της δουλειάς τους είναι το ακόλουθο θεώρημα.

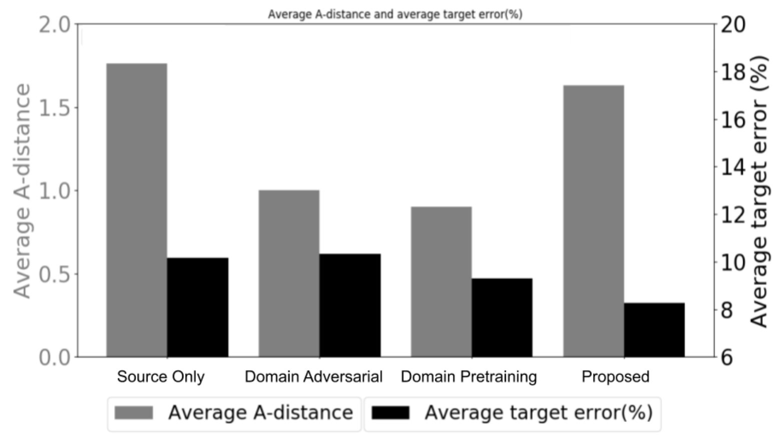
Θεώρημα [12, 10] Έστω H το hypothesis space και D_S, D_T τα δύο πεδία και ϵ_S, ϵ_T οι αντίστοιχες συναρτήσεις λαθών. Τότε για κάθε $h \in H$:

$$\epsilon_T(h) \leq \epsilon_S(h) + \frac{1}{2} d_{H\Delta H}(D_S, D_T) + C \tag{0.4}$$

όπου $d_{H\Delta H}(D_S, D_T)$ [43] είναι το $H\Delta H$-divergence μεταξύ των πεδίων.

Το παραπάνω θεώρημα ορίζει ένα άνω όριο για το εκτιμώμενο σφάλμα στο πεδίο στόχο $\epsilon_T(h)$ για μία υπόθεση h. Αυτό το άνω όριο είναι το άθροισμα τριών όρων, του εκτιμώμενου σφάλματος στο πεδίο πηγή $\epsilon_S(h)$, το divergence μεταξύ των δύο πεδίων $\frac{1}{2} d_{H\Delta H}(D_S, D_T)$ και του σφάλματος της ιδανικής κοινής υπόθεσης C. Η ιδανική κοινή υπόθεση, είναι η υπόθεση εκείνη που ελαχιστοποιεί το άθροισμα του εκτιμώμενου σφάλματος στο πεδίο πηγή και στο πεδίο στόχο. Η ύπαρξη της στην ανίσωση υποδηλώνει πως προκειμένου να πετύχουμε μία προσαρμογή στο πεδίο στόχο από το πεδίο πηγή, πρέπει να υπάρχει μία υπόθεση που πηγαίνει σχετικά καλά και στα δύο πεδία. Όταν μία τέτοια υπόθεση υπάρχει, ο σχετικός όρος θεωρείται σχετικά μικρός και στην πράξη αγνοείται. Ο πρώτος όρος φράσει το εκτιμώμενο σφάλμα στο πεδίο πηγή, αναμένουμε να είναι σχετικά μικρός όσο μαθαίνουμε από τα δεδομένα με επισημείωση του πεδίου. Ο δεύτερος όρος, δίνει μία αίσθηση απόστασης μεταξύ των δύο πεδίων και είναι αυτός που το domain adversarial training προσπαθεί να ελαχιστοποιήσει.

0.7.2 A-distance



Σχήμα 0.4: Σύγκριση του μέσου A-distance και του μέσου σφάλματος για διάφορες μεθόδους προσαρμογής.

Σύμφωνα με το [12] το $H\Delta H$-divergence μπορεί να προσεγγιστεί από το proxy A-distance, το οποίο ορίζεται από την εξίσωση Equation 0.5 δεδομένου του σφάλματος ταξινόμησης των πεδίων e.

$$d_A = 2(1 - 2\min\{e, 1 - e\}) \tag{0.5}$$

Υπολογίζουμε μία προσέγγιση της απόστασης μεταξύ των πεδίων. Ακολουθώντας την βιβλιογραφία δημιουργούμε έναν ταξινομητή SVM που ταξινομεί τα δείγματα με βάση το πεδίο από το οποίο προέρχονται. Εισάγουμε στον ταξινομητή τις αναπαραστάσεις του token $[CLS]$ του BERT, μετράμε το σφάλμα ταξινόμησης και υπολογίζουμε το A-distance με βάση τον τύπο. Εκπαιδεύουμε τον ταξινομητή σε 2000 δείγματα από κάθε πεδίο. Στο Σχήμα 0.4 παραθέτουμε το A-distance, κατά μέσο όρο για τα δώδεκα σενάρια προσαρμογής, χρησιμοποιώντας αναπαραστάσεις που εξήχθησαν από 4 μεθόδους: (1) BERT στα δεδομένα πεδίου πηγής (2) εκτεταμένη προεκπαίδευση πεδίου (Ext. PT), (3) domain adversarial training και (4) την προτεινόμενη μέθοδο. Το domain adversarial trained BERT ελαχιστοποιεί την απόσταση μεταξύ των πεδίων. Η εκτεταμένη προεκπαίδευση πεδίου επίσης μειώνει σημαντικά το A-distance, χωρίς να είναι σχεδιασμένο ώστε σκόπιμα να το πετύχει. Παρατηρούμε πως η προτεινόμενη μέθοδος παρότι ελαχιστοποιεί το σφάλμα δεν μειώνει το A-distance. Μπορούμε να υποθέσουμε πως το BERT έχει την δυνατότητα να επιλύει εξίσου την επιθυμητή εργασία στο πεδίο στόχο και να επιτυγχάνει τον διαχωρισμό των δειγμάτων με βάση το πεδίο προέλευσης. Συνεπώς, χαμηλότερη απόσταση μεταξύ των πεδίων, που επιτυγχάνεται σκόπιμα ή όχι, δεν εγγυάται καλύτερη απόδοση του μοντέλου στο πεδίο στόχο. Επιπρόσθετα, η βελτιστοποίηση της απόδοσης στο πεδίο στόχο δεν απαιτεί ελαχιστοποίηση της απόστασης μεταξύ των πεδίων. Τελικά, δεν παρατηρούμε καμία συσχέτιση μεταξύ του A-distance και της απόδοσης του μοντέλου στο πεδίο στόχο.

0.7.3 Αστάθεια του Domain Adversarial Training

Το domain adversarial training [30] αντιμετωπίζει ορισμένους σημαντικούς περιορισμούς, που καθιστούν την μέθοδο δύσκολη στο να αναπαραχθεί και ασταθή. Για ένα μοντέλο που εκπαιδεύεται με domain adversarial training, με παραμέτρους θ, όπου L_{CLF} η συνάρτηση κόστους ταξινόμησης και L_{ADV} η συνάρτηση κόστους ταξινόμησης του πεδίου προέλευσης και το λ_d να είναι ένας παράγοντας στάθμισης, το domain adversarial training περιγράφεται από το κριτήριο ελαχιστοποίησης της εξίσωσης 1.6.

$$\min_{\theta} L_{CLF}(\theta; D_S) - \lambda_d L_{ADV}(\theta; D_S, D_T) \tag{0.6}$$

Ο δεύτερος όρος της εξίσωσης περιγράφει μία adversarial μεγιστοποίηση της συνάρτησης κόστους ταξινόμησης του πεδίου προέλευσης L_{ADV}. Κανείς μπορεί εύκολα να παρατηρήσει πως η μεγιστοποίηση του L_{ADV} μπορεί να επιτευχθεί απλώς προβλέποντας λάθος πεδίο για κάθε δείγμα που δίνεται στο μοντέλο. Η επιτυχής πρόβλεψη του λάθους πεδίου, δεν καθιστά τα χαρακτηριστικά που μαθαίνει το μοντέλο ανεξάρτητα από το πεδίο προέλευσης, αφού το μοντέλο είναι σε θέση να διαχωρίζει τα πεδία προέλευσης με μία απλή αλλαγή των ονομάτων των δύο πεδίων. Εμπειρικά παρατηρήσαμε αυτή την συμπεριφορά όταν προσπαθήσαμε να εκπαιδεύσουμε το BERT με domain adversarial training.

Οι Shu κ.ά. [88] αντιμετωπίζουν τυπικά το ζήτημα, αποδεικνύοντας πως όταν ένα μοντέλο f έχει θεωρητικά άπειρη χωρητικότητα, και τα δύο πεδία είναι διακριτά, τότε το μοντέλο μπορεί να επιβάλλει αυθαίρετους μετασχηματισμούς στο πεδίο στόχο, ώστε να ταιριάζει με την κατανομή του πεδίου πηγή.

0.8 Συμπεράσματα και Μελλοντικές Προεκτάσεις

Σε αυτήν την εργασία διερευνούμε το πρόβλημα του unsupervised domain adaptation για sentiment analysis και προτείνουμε μια νέα μέθοδο για την αντιμετώπιση του ζητήματος. Διαπιστώσαμε ότι η δημοφιλής τεχνική domain adversarial training αντιμετωπίζει θεωρητικούς και εμπειρικούς περιορισμούς στην εποχή των προεκπαιδευμένων γλωσσικών μοντέλων υψηλής χωρητικότητας, οδηγώντας σε αστάθειες εκπαίδευσης και σε χειροτέρευση της απόδοσης για την εργασία που μελετήθηκε. Προτείνουμε μια μέθοδο, η οποία αποτελείται από δύο βήματα, εκτεταμένη προ-εκπαίδευση στο domain στόχο και fine-tuning με MLM. Τα πειράματα στο σύνολο δεδομένων των κριτικών Amazon, αποδίδουν βελτιωμένα αποτελέσματα, που ξεπερνούν τις προηγμένες μεθόδους από όλες τις καθιερωμένες προσεγγίσεις.

Η προτεινόμενη μέθοδος θα μπορούσε ακόμα να εφαρμοστεί σε άλλες εργασίες όπως το question answering και το part-of-speech tagging. Ένα άλλο πεδίο εφαρμογής θα μπορούσε να είναι η γλωσσικές μεταβολές στον χρόνο και σε στυλ. Θα ήταν πολύ σημαντικό να διερευνήσουμε περαιτέρω σχετικές βοηθητικές εργασίες τόσο για την προ-εκπαίδευση όσο και για βελτιστοποίηση, καθώς και τρόπους συνδυασμού όλων των losses. Στο μέλλον θα θέλαμε να διερευνήσουμε τη δυνατότητα εφαρμογής της μεθόδου μας σε επιβλεπόμενα σενάρια, όπου η βοηθητική εργασία MLM θα μπορούσε να χρησιμοποιηθεί σε δεδομένα χωρίς επισημείωση που προέρχονται από το ίδιο ή παρόμοιο πεδίο.

Chapter 1

Introduction

1.1 Motivation

Deep architectures have achieved state-of-the-art results in a variety of machine learning tasks. While the vast majority of deep learning systems are trained and evaluated on a specific data distribution, real word models are often used in out-of-domain settings which results in performance degradation. Collecting and annotating data to train specific models is a costly and time-consuming endeavor and hinders the re-usability of trained models, while unlabeled data are easily accessible. Unsupervised Domain Adaptation (UDA) is an active research area with high impact in real world adaptation of machine learning models.

Recent advances in Language Processing are leaded by pretrained language models, trained on massive general corpora, with various sources, like web content, Wikipedia, news articles and literary works. Representations learned by such models achieve strong performance across a great variety of tasks and data from a variety of sources. Those advances create a question on the relevance of task text domain when learning a new task. As domain is typically called a data distribution over language that characterizes a topic, genre or class of data. In particular a question of interest is if such systems are able to be fine-tuned for a specific task and target domain, from data belonging to a similar but distinct source domain.

As far as domains are relevant, and given the wide variety of domains that human language presents, a need arises to collect and curate data for each domain of interest. If a machine learning system is designed to be used across a wide range of domains, the effort to annotate data for each domain may be a limitation, especially as language evolves over time.

Moreover, changes in data distributions, between training and deployment data, is a high concerning issue for machine learning algorithms deployed for production. A model is in general expected to perform well on examples that coming from a similar distribution, outside of the labeled training set.

Natural Language Processing usually suffers from the lack of labeled data in the exact domain of a certain application. In contrast, unlabeled corpora, are in general accessible via retrieval from the web. Therefore pretrained language models, is a good fit in order to prepare models for language processing, as they not require labeled training data, except little task-specific data for fine-tuning.

1.2 Research Objectives & Contribution

Domain adaptation is the ability to apply an algorithm trained in one or more source domains to a different (but related) target domain. Domain adaptation is a subcategory of transfer learning. In domain adaptation, the source and target domains all have the same feature space (but different distributions); in contrast, transfer learning includes cases where the target domain's feature space is different from the source feature space. The present study is concerned with the question of whether domains are still a limitation for natural language and how to develop modern machine learning methods that successfully adapt to domains.

The main research objective of this work is to develop an approach to unsupervised domain adaptation for natural language processing applications, that is inspired and well fitted to the recent cutting edge developments to the field. That is achieving adaptation to a language target domain by exploiting the power of large, transformer based, pretrained language models and the appropriate usage of the available unlabeled data.

We review the literature on unsupervised domain adaptation for natural language processing, considering the approaches before the establishment of pretrained models, as also the recent advances that bring well established methods on the context of pretrained language models. We found the well-spread domain adversarial technique facing theoretical and empirical limitations in the era of high-capacity transformer based pretrained language models, leading to training instabilities and to no improvement for the studied task.

The main contributions are: (a) We propose a novel, simple and robust unsupervised domain adaptation procedure for downstream BERT models based on multitask learning, (b) we achieve state-of-the-art results for the Amazon reviews benchmark dataset, surpassing more complicated approaches and (c) we conduct a discussion on the limitations of adversarial domain adaptation, grounded on theoretical concepts and our empirical observations.

In this work we develop a simple and effective approach to unsupervised domain adaptation, that leverages the capabilities of the high-capacity, transformer-based, pretrained models, such as in our case BERT [24]. Our method is based on simultaneously learning the task from labeled data in the source distribution, while adapting to the language in the target distribution using multitask learning. The key idea of our method is that by simultaneously minimizing a task-specific loss on the source data and a language modeling loss on the target data during fine-tuning the model will be able to adapt to the language of the target domain, while learning the supervised task from the available labeled data. Starting from a pretrained in general corpora BERT model, we continue the pretraining Masked Language Modeling on relevant and relevantly small amount of unlabeled target domain data. During fine-tuning, while learning the task from the labeled source data, we keep the MLM objective on the target domain data, in multi-tasking manner.

Additionally, we conducted some studies relevant to our main work about the sample efficiency of our method and on stopping criteria under unsupervised domain adaptation. We investigate the impact of using different amount of target domain unlabeled data on model performance, to study the sample efficiency of our model. A common problem when performing UDA is the lack of target labeled data that can be used for hyperparameter validation. We explore how validation can be performed when no labeled target data are available and propose the minimization of the mixed training objective as an effective stopping criterion.

Finally, an important part of that work concerns the domain adversarial training that is based on the provided theory for learning from many domains [12, 11] and the relationship of the theory with our empirical results on the studied task. We examine if minimizing the distance between domains is sufficient in order to adapt to the target domain and discuss the limitations and training instabilities of the dominant domain adversarial training.

1.3 Thesis Outline

The thesis is structured as follows:

In Chapter 2 we provide the machine learning background. Specifically, we first introduce the basic concepts of machine learning and then focus on the deep learning background, that is most relevant to our work. We also provide insights on basic machine learning concepts and techniques.

In Chapter 3 we discuss main concepts of Natural Language Processing. We present the historical progress on words representations, embeddings, language modeling and finally BERT, Bidirectional Encoder Representations from Transformers, that is the base for our approach to unsupervised domain adaptation.

An extensive literature review of unsupervised domain adaptation in Natural Language Processing is provided in Chapter 4. We explore the basic concepts over the most well spread approaches such as model based, including pivo-based adaptation, loss based, including the dominant in practice domain adversarial training, and data-based approaches that include psuedo-labeling algorithms and the usage of pretrained language models.

Our main research work is included in Chapter 5. The Chapter includes the motivation behind our approach to unsupervised domain adaptation, an analysis of the main proposed method, experimental results on the multi-domain Amazon reviews dataset, a theoretical approach and comparison with related previous methods and finally a discussion on the limitations of domain adversarial training.

Finally, Chapter 6 includes conclusions inferred from the thesis and provides an outlook into the future.

Chapter 2

Machine Learning

2.1 Defining Machine Learning

Machine Learning (ML) is the field of Artificial Intelligence (AI) that studies computer algorithms that improve automatically through experience. Machine Learning algorithms create mathematical models based on sampled data in order to make predictions or decisions without being explicitly programmed to do so. Machine Learning algorithms are used in a wide variety of applications, when is challenging for a human to manually design the needed algorithms, such as in computer vision and natural language processing.

The classic machine learning procedure follows the scientific paradigm of induction and deduction. In the inductive step we learn the model from raw data (so called training set), and in the deductive step the model is applied to predict the behaviour of new data.

An example is a pair in the form $(x, f(x))$, where x is the input and $f(x)$ is the output, or the value of the function, or the image of x by f. Pure inductive inference or induction can be described as follows: Given a set of examples of f, find a function h that approaches f. The function h is called hypothesis. The reason why learning is difficult from a conceptual point of view, is that is difficult to say if a particular function h is a good approach of f. A good hypothesis should generalize, that is it should predict correctly unseen examples.

2.2 Machine Learning approaches

Machine Learning approaches are usually divided into three broad categories, depending on the feedback available to the learning algorithm: supervised learning, unsupervised learning and reinforcement learning. Other approaches have also developed that do not fit into this categorisation and some systems use more than one approach, such as in the semi-supervised learning.

Supervised Learning The supervised learning problem involves learning a function from input and output example pairs. The system is presented with example inputs and their desired outputs also called labeled training data, given by a teacher, and the goal is to learn a general rule that maps inputs to outputs. A supervised learning algorithm analyzes the labeled training data and produces an inferred function, which can be used to map new input examples, and in the optimal scenario will correctly determine the output for unseen instances.

Unsupervised Learning The unsupervised learning problem involves learning input patterns without given output values (labels). Also known as self-organization, unsupervised learning allows for modeling of probability densities over inputs.

Semi-supervised Learning Semi-supervised learning is an approach to ML that combines a small amount of labeled data with a large amount of unlabeled data during training. Semi-supervised learning combines the unsupervised (with no labeled training data) and supervised learning (with only labeled training data) approaches. Unlabeled data, when used along with a small amount of labeled data, can improve learning performance. Collecting labeled data for a learning problem often requires skilled humans, or a physical experiment. That cost associated with the labeling process may make large, fully labeled training sets unachievable, while acquisition of unlabeled data is relatively inexpensive. Therefor, semi-supervised learning can be of great practical value.

Reinforcement Learning Reinforcement Learning is an approach to Machine Learning concerned with how systems have to take decisions in order to maximize a cumulative reward. Reinforcement learning differs from supervised learning in not needing labelled input/output pairs to be presented and any sub-optimal actions to be explicitly corrected. The problem setting is typically stated in the form of a Markov decision process, as many reinforcement learning algorithms utilize dynamic programming techniques. Reinforcement Learning algorithms differ from classical dynamic programming methods in that the first do not assume knowledge of an exact mathematical model of the Markov decision process and they target such processes where exact methods become impracticable.

2.3 Perceptron, Activations and Feedforward Networks

2.3.1 Perceptron

The perceptron is an algorithm for supervised learning of binary classifiers. It is a type of linear classifier, a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector. The perceptron is therefore a function that maps its input $\mathbf{x}$ to an output value $f(\mathbf{x})$, a single binary value:

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

where $\mathbf{w}$ is a vector of real weights, $\mathbf{w} \cdot \mathbf{x}$ is the dot product and b is the bias. The value $f(\mathbf{x})$ is used to classify $\mathbf{x}$ as either positive or negative instance, in the case of a binary classification problem.

The perceptron learning algorithm does not terminate if the set is not linearly seperable. A famous example of the perceptron's inability to solve problems with linearly nonseparable vectors is the Boolean exclusive-or problem.

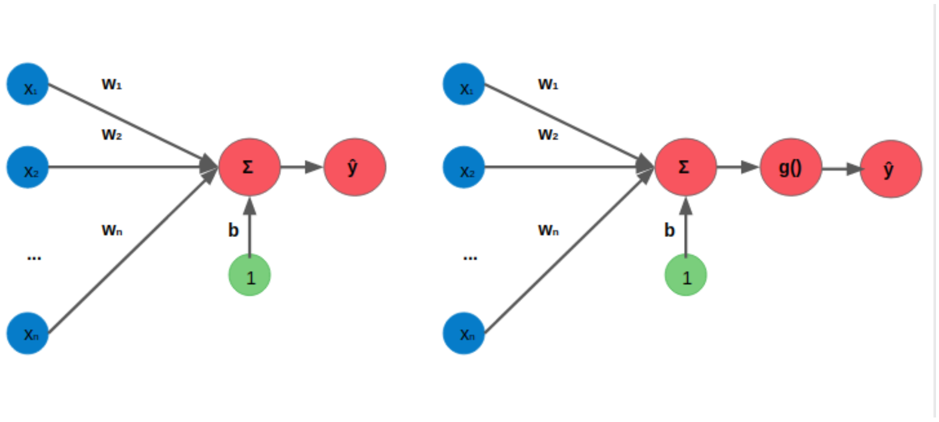


Figure 2.1: Left: a simple perceptron unit. Right: a percpetron unit with an addition of an activation function.

2.3.2 Activation Functions

In order to solve linearly nonseperable problems, it is essential to introduce non-linearities into the algorithms. Non-linearities allow us to approximate arbitarily complex functions. We can achieve such results with activation functions. Activation functions define the output of a node, given the inputs, and perform the essential task of making a non-linear decision. Common activation functions, introduce a first non-linearity at zero and some, like sigmoid, use a second non-linearity for large inputs. Given a perceptron as in Equation (2.1) and an activation function g we can define a simple classifier with one node as follows:

$$\hat{y} = g(\mathbf{w} \cdot \mathbf{x} + b) \tag{2.2}$$

A simple perceptron unit and a perceptron unit with an activation function is shown in Figure 2.1.

Some common activation functions follow:

Sigmoid A sigmoid function has a characteristic “S”-shaped curve, also known as sigmoid curve. A common example of a sigmoid function is defined by the formula:

$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} \tag{2.3}$$

A sigmoid function is a bounded, differentiable, real function that is defined for all real input values and has a non-negative derivative at each point. In general, a sigmoid function is monotonic, and has a first derivative which is bell shaped. A sigmoid function is constrained by a pair of horizontal asymptotes as $x \to \pm\infty$. The sigmoid function is convex for values less than 0, and it is concave for values more than 0. Because of this, the sigmoid function and its affine compositions can possess multiple optima.

However, sigmoid as defined by Equation (2.3) has two disadvantages. First of all when its value is close to 0 or 1, the gradient value is close to 0. Secondly, the sigmoid output is not centered at 0. So, if the data coming to the neuron has always positive values, the gradient of weights will be either always positive, or always negative and an unwanted alternation of them is introduced into the network.

Hyperbolic tangent Hyperbolic tangent is defined by the formula:

$$\text{tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.4}$$

The tanh function is a scaled and shifted sigmoid. From Equations (2.3) and (2.4) we can get the relation $\text{tanh}(x) = 2\sigma(2x) - 1$. So, the hyperbolic tangent is a bounded, differentiable, real function that is defined for all real input values and has a non-negative derivative at each point. The advantage of tanh over the sigmoid defined by Equation (2.3) is that tanh is zero-centered. Nevertheless, the disadvantage of the gradient value close to 0 and 1 remains.

Rectified Linear Unit (ReLU) The rectifier is an activation function defined as the positive part of its argument:

$$f(x) = x^+ = \text{max}(0, x) \tag{2.5}$$

This is also known as a ramp function and is analogous to half-wave rectification in electrical engineering. A main advantage of ReLU is the sparce activation, as in a randomly initialized network, only about 50 % of hidden units are activated (have a non-zero output). Other major advantages are a better gradient propagation, efficient computation and the scale invariant property. Potential disadvantages of ReLU are the lack of differentiability at zero and zero-center symmetry and the fact that is unbounded. A well known disadvantage is the so called “dying ReLU” problem, during which neurons are pushed into states that they become inactive for essentially all inputs, no gradients flow backward through the neuron, and so the neuron becomes stuck in a perpetually inactive state.

Leaky ReLU In order to address the "dying ReLU" problem, the leaky ReLU is defined as:

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ ax & \text{otherwise.} \end{cases} \tag{2.6}$$

where a is a small constant, e.g. $a = 0.1$. So, leaky ReLU gives a small negative value for negative inputs.

The above mentioned activation functions, sigmoid, $tanh$, ReLU and leaky ReLU, are shown in Figure 2.2.

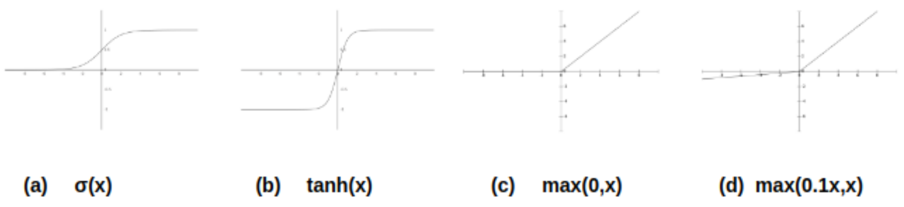


Figure 2.2: Activation functions. (a) The sigmoid function. (b) The tanh function. (c) The ReLU function. (d) The leaky ReLU function with $a = 0.1$.

2.3.3 Feedforward Networks

Multilayer perceptrons (MLPs), also called feedforward networks, consist of multiple stacked layers of perceptron units, which are connected without any feedback loops. Each perceptron acts as a computational unit that takes as input the output of the previous layer and implements a function that converts the input vector to a scalar. Typically units of a single layer are not connected to each other. A feedforward network consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Such a model is made of an input layer that accepts the input data, hidden layers that process outputs from the previous layer and an output layer that provides the final output. The flow of information from the input to the output is called forward propagation. The number of hidden layers determines the depth of a model, while the number of the units in the hidden layer determines the width of the model. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron, as it can distinguish data that is not linearly separable. In Figure 2.3 is visualized the difference between a simple neural network and a deep neural network.

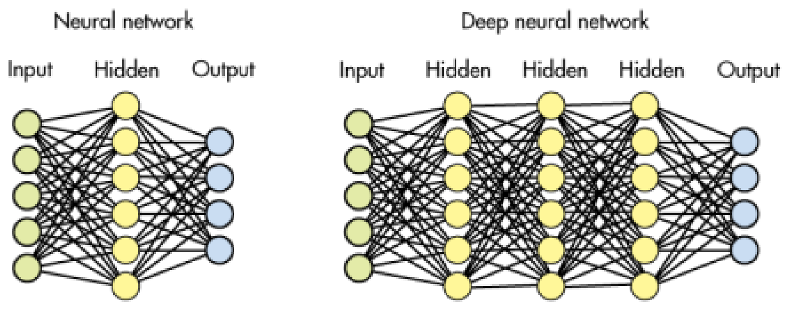


Figure 2.3: A neural network and a deep neural network, with more than 3 hidden layers. Both consisting of input, hidden and output layers.

2.4 Training

2.4.1 Quantify Loss

Artificial Neural Networks are trained by an optimization method, that aims to select a set of model parameters that minimizes the prediction error. The prediction error of a model f with parameters w is estimated by a loss function $J(w)$. The loss function computes a non-negative value that measures the inconsistency between the predicted and the target output.

$$\mathcal{L}(f(x^{(i)}; w), y^{(i)}) \tag{2.7}$$

By applying Equation 2.7 on the entire dataset, we can quantify the total loss over the dataset, also known as the objective function, cost function or empirical risk.

$$J(w) = \frac{1}{n} \sum_{i=0}^{n} \mathcal{L}(f(x^i; w), y^{(i)}) \tag{2.8}$$

Binary Cross Entropy Loss For models that output a probability between 0 and 1, a good fit as a loss function is the binary cross entropy.

$$J(w) = \frac{1}{n} \sum_{i=0}^{n} y^{(i)} \log(f(x^{(i)}; w)) + (1 - y^{(i)}) \log(1 - f(x^{(i)}; W)) \tag{2.9}$$

Mean Squared Error Loss For regression models that output continuous real numbers, mean squared error loss can be used.

$$J(w) = \frac{1}{n} \sum_{i=0}^{n} (y^{(i)} - f(x^{(i)}; W))^2 \tag{2.10}$$

Loss Optimization Training an artificial neural network is in fact the optimization problem of finding a set of model parameters w that minimizes the objective function $J(w)$ over the training data. We want to find the network weights that achieve the lowest loss.

$$\begin{aligned} w^* &= \underset{w}{\text{argmin}} \frac{1}{n} \sum_{i=0}^{n} \mathcal{L}(f(x^i; w), y^{(i)}) \\ &= \underset{w}{\text{argmin}}\, J(w) \end{aligned} \tag{2.11}$$

2.4.2 Gradient Descent

The above loss optimization problem is usually addressed by the gradient descent algorithm. Gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. To find a local minimum of a function using gradient descent, we take steps proportional to the negative of the gradient, or approximate gradient, of the function at the current point. Gradient descent is based on the observation that if a multi-variable function $f(x)$ is defined and differentiable in a neighborhood of a point a, then $f(x)$ decreases fastest if one goes from point a in the direction of the negative gradient of f at a, $-\nabla f(a)$. It follows that the next point could be chosen as $a_{n+1} = a_n - \gamma \nabla f(a_n)$ for γ a positive real, small enough, and then it will stand that $f(a_n) \geq f(a_{n+1})$. We subtract $\gamma \nabla f(a)$ from a in order to move against the gradient, towards the local minimum. Given this, we can generalize by starting with a random point x_0 as a guess for a local minimum of f and to obtain a sequence $x_0, x_1, x_2, ...$ such that $x_{n+1} = x_n - \gamma_n \nabla f(x_n), n \geq 0$. We then get the monotonic sequence $f(x_0) \geq f(x_1) \geq f(x_2) \geq$ ..., so the sequence (x_n) converges to the desired local minimum. When f is convex, all local minima are also global minima, so in this case gradient descent can converge to the global solution.

With certain assumptions on the function f, such as f being convex and ∇f being Lipschitz, and particular choices of γ, convergence to a local minimum can be guaranteed. Unfortunately, deep neural networks non-linearity causes the loss surface to become non-convex. This means that there is no guarantee that a gradient descent based method will converge to a global minimum.

Algorithm 1 Gradient Descent

```
Initialize weights $w$ randomly $\sim \mathcal{N}(0, \sigma^2)$
repeat
  Compute gradient, $\frac{\partial J(w)}{\partial w}$
  Update weights, $w \leftarrow w - \gamma \frac{\partial J(w)}{\partial w}$
until convergence
return weights $w$
```

Learning Rate The hyper-parameter γ used in Gradient Descent algorithm, that scales the gradient in order to keep each step small enough, is called learning rate. The gradient defines the direction in which we should update the weights, but it does not provide any information about the magnitude of the update step. This, the size of each iteration step toward a minimum, is controlled by the learning rate. Since it determines to what extent newly acquired information overrides old information, it represents the speed at which a model learns. According to the chosen rate, there is a trade-off between convergence and overshooting. Small learning rates converge slowly and gets stuck in false local minima. Large learning rates jump over minima, become unstable and diverge.

In order to deal with the learning rate decision problem, achieve faster convergence, prevent oscillations and getting stuck in undesirable local minima the learning rate is often varied during training. One way of achieving so, is scheduling the rate based on time or step of training or in an exponential manner. Another way is by choosing an adaptive learning rate algorithm that calculates it depending on how large gradient is, how fast learning is happening, size of particular weights etc. Some famous adaptive gradient descent algorithms are Adam, Adadelta, Adagrad and RMSprop.

Backpropagation In order to use Gradient Descent algorithm to address the loss optimization problem of deep neural networks, is essential to compute the gradient of the loss function with respect to the weights of the network. Backpropagation is a widely used algorithm in machine learning, that efficiently computes the desired gradient of the loss function with respect to the weights of the network for a single input-output example. The efficiency of the algorithm makes it feasible to use gradient descent for training multi-layer networks. Backpropagation works by computing the gradient of the function with respect to each weight by the chain rule, computing the gradient one layer at a time, iterating backward from the last layer to avoid redundant calculation, in a dynamic programming way.

Let f be the network, with L layers, $W^l = (w^l_{jk})$ being the weights between layer $l-1$ and l, where w^l_{jk} is the weight between the k^{th} node in layer $l-1$ and the j^{th} node in layer l, g^l is the activation function at layer l, x is an input instance and y the target output. The overall model is a combination of function composition and matrix multiplication and can be written as:

$$f(x) = g^L(W^L f^{L-1}(W^{L-1} ... f^1(W^1 x))) \tag{2.12}$$

Given the input - output pair the loss of the model is:

$$J(W) = \mathcal{L}(f(x^{(i)}; W), y^{(i)}) \tag{2.13}$$

Backpropagation computes the gradient for a fixed input–output pair, where the weights can vary. Each component of the gradient $\frac{\partial J}{\partial w^l_{jk}}$, can be computed by the chain rule, but doing so for each weight is inefficient. Backpropagation computes the gradient by avoiding duplicate calculations and not computing unnecessary intermediate values, by computing the gradient of each layer – specifically, the gradient of the weighted input of each layer.

Mini batch Stochastic Gradient Descent A huge challenge according gradient descent, also known as batch gradient descent, is the fact that the gradient computation over the whole dataset can be very computationally intensive. To address the issue, stochastic gradient descent was introduced, that proposes to pick on each iteration a single data point i, and update the parameters using the gradient of the loss of just that datum, $\frac{\partial J_i(w)}{\partial w}$. Although it is faster than batch gradient descent, it can lead to noisy (stochastic) gradients and cause the loss function to fluctuate.

In practice a third version of the algorithm is used that combines batch and stochastic variants, named mini batch stochastic gradient descent. A mini batch of B data points are picked and the average gradient over those B points is calculated and used, which is fast to compute and a much better estimate of the true gradient. The more accurate estimation of the gradient, makes convergence smoother and allows for larger learning rates. Moreover the selection of mini-batches lead to fast training, as the computation can be parallelized and GPU usage enables significant speed increases.

The size of a batch B, known as batch size, is a learning hyper parameter of high importance that is set empirically. Larger batches provide a more accurate estimate of the gradient, while noisier estimates of small batches can help by preventing overfitting. The batch size is often limited by the available memory of the hardware.

Algorithm 2 Stochastic Gradient Descent

```
Initialize weights $w$ randomly $\sim \mathcal{N}(0, \sigma^2)$
repeat
  Pick single data point $i$
  Compute gradient, $\frac{\partial J_i(w)}{\partial w}$
  Update weights, $w \leftarrow w - \gamma \frac{\partial J_i(w)}{\partial w}$
until convergence
return weights $w$
```

Algorithm 3 Mini-batch Stochastic Gradient Descent

```
Initialize weights $w$ randomly $\sim \mathcal{N}(0, \sigma^2)$
repeat
  Pick batch of $B$ data points
  Compute gradient, $\frac{\partial J(w)}{\partial w} = \frac{1}{B} \sum_{k=1}^{B} \frac{\partial J_k(w)}{\partial w}$
  Update weights, $w \leftarrow w - \gamma \frac{\partial J(w)}{\partial w}$
until convergence
return weights $w$
```

2.5 Recurrent Neural Networks

Modeling Sequences A common machine learning task is modeling data that appear in the form of sequences, such as audio and text. Modeling sequential data can be demanding given dependencies, such as lengths, relevant positions and the order of the features. Therefore in order to model sequences we need to handle variable lengths, track long-term dependencies, maintain information about order and share parameters across the sequence.

2.5.1 Vanilla RNNs

Recurrent Neural Networks (RNNs) are a type of neural networks that process sequences of data. Unlike feedforward networks that operate under the assumption that all training instances independent, RNNs produce their output taking into consideration information previously presented to them. The core idea of an RNN is to infer the temporal dynamics of the data sequence by keeping an internal state, also known as hidden layer, and updating it on each time-step, for each new datum of the sequence.

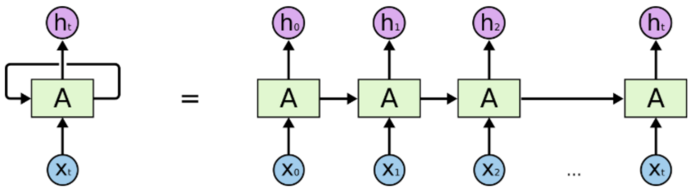


Figure 2.4: A RNN unit as loop over time steps and unfolded over time. Figure from [64].

Figure 2.4 introduces the RNN architecture where each box is the hidden layer at a time-step t, h_t. Each layer performs a linear matrix operation on its inputs followed by an non-linear operation. At each time-step, there are two inputs to the hidden layer: the output of the previous time-step h_{t-1} and the input at the current time-step x_t. Each input is multiplied by a dedicated weight matrix W^{hh} and W^{hx} to produce the cell hidden state h_t. The latter is multiplied with a weight matrix W^s to obtain the output prediction $\hat{y}$.

$$h_t = f(W^{hh}h_{t-1} + W^{hx}x_t) \tag{2.14}$$

$$\hat{y}_t = W^s h_t \tag{2.15}$$

The above Equations (2.14) and (2.15) describe the functionality of a RNN cell, that computes the hidden layer and output features at each time-step, when the following applies:

- x_1, ..., x_t, ..., x_T: Sequence of T input vectors, e.g. a sentence of T words.
- $x_t \in \mathbb{R}^d$: input vector at time-step t.
- $W^{hx} \in \mathbb{R}^{d_h \times d}$: weights matrix used to condition the input vector x_t.
- $W^{hh} \in \mathbb{R}^{d_h \times d_h}$: weights matrix used to condition the output of the previous time-step h_{t-1}.
- $h_{t-1} \in \mathbb{R}^{d_h}$: hidden layer at the previous time-step, $t-1$.
- $h_0 \in \mathbb{R}^{d_h}$: initialization vector for the hidden layer when $t=0$.
- $f()$: non-linearity function.
- $\hat{y}_t \in \mathbb{R}^l$: the output at time-step t.
- $W^s \in \mathbb{R}^{d_h \times l}$: weights matrix used to take an output $\hat{y}_t$.

We should underline that the same weights W^{hh} and W^{hx} are applied repeatedly at each time-step. So, the number of parameters of the model is less, and independent of the length of the input.

Advantages and disadvantages of RNNs RNNs have several advantages, that makes them suitable for various applications. They can process input sequences of any length. The model size does not increase for longer input sequence lengths. Computation for each step uses in theory information from all previous steps. Same weights are applied to every time-step of the input, so there is symmetry in how inputs are processed. On the other hand computation is slow, as it is sequential it cannot be parallelized. Moreover, in practice, information from many steps back is difficult to access.

2.5.2 Exploding and Vanishing Gradient Problems

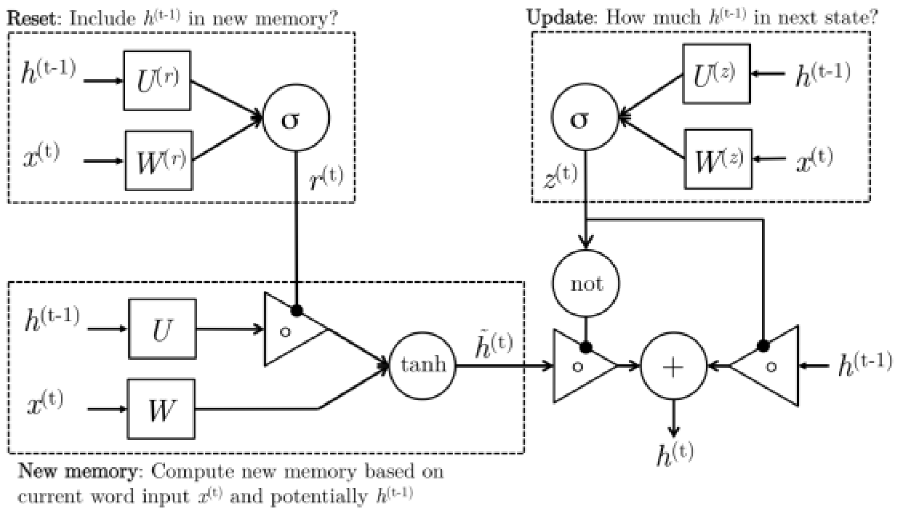
Recurrent neural networks propagate weight matrices from one time-step to the next, while the goal is to enable propagating context information through faraway time-steps. As described in previous section 2.4, in order to train the network, gradients of the loss with respect to each parameter should be calculated. Computing the gradient with respect to h_0 involves many factors of W^{hh}.

If many of the intermediate values are greater than one, then the gradient value grows extremely large and it causes an overflow at training runtime. The issue is called gradient explosion problem. To solve the problem a simple heuristic solution is used, that clips gradients to small number whenever they explode. That is, whenever they reach a certain threshold, they are set back to a small number.

On the contrary when many values on the computational chain are between zero and one, the multiplication of many such small numbers together causes smaller gradients tending to zero and makes further back time steps irrelevant for the hidden state. That is the vanishing gradients problem, due to which we cannot distinguish if there is no dependency between two steps in the data or the dependency is not captured. A usual solution to the problem is the usage of ReLU as activation function. ReLU's derivative prevents the gradients from shrinking, as it is either 0 or 1. Another technique to face vanishing gradients is that instead of initializing W^{hh} and biases randomly, setting W^{hh} equal to the identity matrix and biases to zero.

2.5.3 Gated Recurrent Units (GRU)

Taking into account the above issues, RNNs have been found to perform better with the use of more complex units for activation, that control what information is passed through. Gated recurrent units are designed in a manner to have more persistent memory, that makes it easier for the model to capture long-term dependencies. A Gated Recurrent Unit (GRU) can be mathematically described with the following four Equations, that correspond to GRU's core operational stages. A detailed layout of a GRU can be found on Figure 2.5.



Reset: Include $h^{(t-1)}$ in new memory?
$h^{(t-1)}$
$U^{(r)}$
$x^{(t)}$
$W^{(r)}$
σ
$r^{(t)}$
Update: How much $h^{(t-1)}$ in next state?
$U^{(z)}$
$h^{(t-1)}$
σ
$W^{(z)}$
$x^{(t)}$
$z^{(t)}$
not
$h^{(t-1)}$
U
$x^{(t)}$
W
tanh
$\tilde{h}^{(t)}$
+
$h^{(t-1)}$
$h^{(t)}$
New memory: Compute new memory based on current word input $x^{(t)}$ and potentially $h^{(t-1)}$

Figure 2.5: The detailed internal of a GRU. Figure from [57]

$$z_t = f(W^z x_t + U^z h_{t-1}) \tag{2.16}$$

$$r_t = f(W^r x_t + U^r h_{t-1}) \tag{2.17}$$

$$\tilde{h_t} = \tanh(r_t \circ U h_{t-1+W x_t}) \tag{2.18}$$

$$h_t = (1 - z_t) \circ \tilde{h_t} + z_t \circ h_{t-1} \tag{2.19}$$

A new memory $\tilde{h_t}$ is the combination of the new input x_t with the previous hidden state h_{t-1}. Reset Gate, r_t, determines the importance of h_{t-1} for the new memory $\tilde{h_t}$ and has the ability to completely diminish irrelevant past hidden state. Update gate z_t is responsible for determining the percentage of h_{t-1} to be carried to the next time-step. Hidden state h_t is generated as a combination of the past hidden state h_{t-1} and the new memory $\tilde{h_t}$ on the percentage update gate determines.

2.5.4 Long-Short-Term Memories (LSTM)

Long-Short-Term Memories, commonly known as LSTM, is a complex activation unit that has been used successfully to a wide range of applications. The motivation for using the LSTM architecture is similar to those for GRUs. The mathematical formulation of LSTMs follows, as well as a figure with detailed internals of an LSTM cell.

$$i_t = \sigma(W^i x_t + U^i h_{t-1}) \tag{2.20}$$

$$f_t = \sigma(W^f x_t + U^f h_{t-1}) \tag{2.21}$$

$$o_t = \sigma(W^o x_t + U^o h_{t-1}) \tag{2.22}$$

$$\tilde{c}_t = \tanh W^c x_t + U^c h_{t-1} \tag{2.23}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \tag{2.24}$$

$$h_t = o_t \circ \tanh c_t \tag{2.25}$$

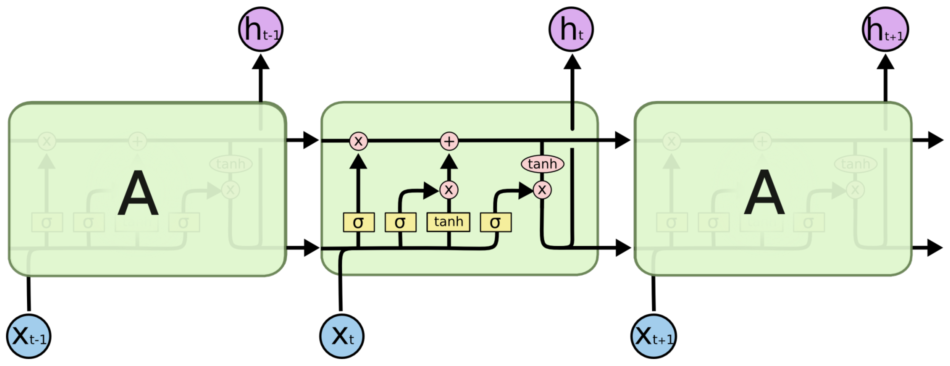


Figure 2.6: The internal of an LSTM. Figure from [64]

Intuition of the structure of an LSTM can be gained by considering the following stages.

- New memory generation, as the similar stage of a GRU, The input x_t and the past hidden state h_{t-1} is used to produce new memory $\tilde{c}_t$ which includes aspects of the new input x_t.

- Input gate function is to check if the new datum is important. The input gate uses the input x_t and the past hidden state h_{t-1} to determine if the input us worth preserving and is used to gate the new memory. It produces i_t as an relevant indicator.

- Forget gate is similar to input gate, in the way that determines if the past memory information is useful for the computation of the current memory. The forget gate takes into consideration input x_t and past hidden state and produces f_t.

- Final memory generation uses the forget gate f_t advice to keep or forget past memory c_{t-1}. Similarly it takes the advice of the input gate i_t and accordingly gates the new memory $\tilde{c}_t$. The addition of those two results produce the final memory c_t.

- Output - Exposure gate purpose is to separate final memory c_t from hidden state h_t. Final memory contains information that is not necessarily required to be saved in the hidden state, so this gate makes the assessment regarding what parts of the memory c_t needs to be exposed in the hidden state.

2.5.5 Bidirectional RNNs

Simple RNNs condition on previous input instances to predict the current output. It is possible to make predictions based on future input instances, by having the RNN model read input backwards. A combination of those approaches is called a bi-directional deep neural network, that at each time-step t maintains two hidden layers, one for the left-to-right propagation and and another for the right-to-left propagation. Such a network, that maintains two hidden layers at any time, consumes twice as much memory space for its parameters. The final result $\hat{y}_t$ is generated through combining the score results produced by both RNN hidden layers.

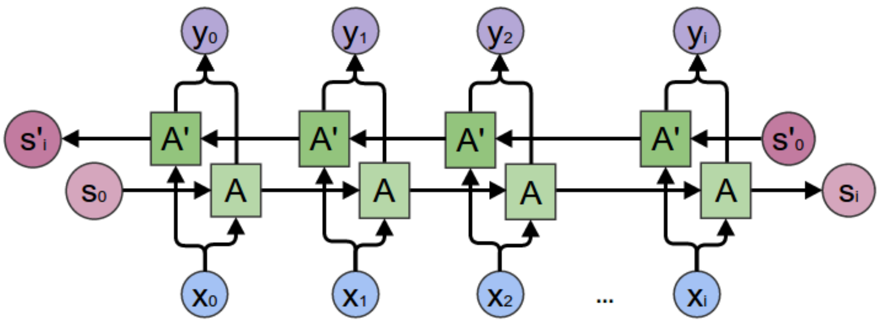


Figure 2.7: A bi-directional RNN. Figure from [64]

In the general case, an RNN can be multi-layered with L layers. Figure 2.7 shows bi-directional RNN. At time-step t each intermediate neuron receives one set of parameters from the previous(or next) time-step (in the same layer), and two sets of parameters from the previous RNN hidden layer, one that comes from the left-to-right RNN and the other from the right-to-left RNN. A bidirectional RNN with L layers can be described by the following Equations (2.26), (2.27), (2.28), (2.29). $\overrightarrow{h_t^i}$ is the left-to-right hidden state at layer i and time-step t, that takes as input the left-to-right state at layer layer $i-1$ and same time-step t and the left-to-right state at the same layer i and left $t-1$ time-step.

$\overleftarrow{h_t^i}$ is the symmetrical for the right-to-left hidden state. At the last layer, $\overrightarrow{h_t^L}$ and $\overleftarrow{h_t^L}$ are combined and the combination of them is used for the final prediction y_t at time-step t.

$$\overrightarrow{h_t^i} = f(\overrightarrow{W^i} h_t^{i-1} + \overrightarrow{V^i} h_{t-1}^i + \overrightarrow{b^i}) \tag{2.26}$$

$$\overleftarrow{h_t^i} = f(\overleftarrow{W^i} h_t^{i-1} + \overleftarrow{V^i} h_{t-1}^i + \overleftarrow{b^i}) \tag{2.27}$$

$$h_t = c(\overrightarrow{h_t^L}, \overleftarrow{h_t^L}) \tag{2.28}$$

$$y_t = g(Uh_t + c) \tag{2.29}$$

2.5.6 Attention Mechanisms

A problem occurred with recurrent neural networks is related with the state used for the next layer of the network, as the last time-step hidden state or an average of all hidden states may not be the right choice for various applications. Attention has been established as valuable tool to face that issue. Originally, attention was used in Neural Machine Translation to dynamically attend over the representation of the input sequence and predict the next word in the translated sequence. Attention can be interpreted as focusing on the most relevant elements, e.g. words, of the input by computing weights of importance for their representations.

The importance of each element is typically measured by their alignment with a query vector q. Given a sequence of N input vectors $x_1, ..., x_N$ the attention module uses an alignment function to score the relevance of each x_i to the query q. The alignment scores s_i are then normalized using the softmax function to produce attention weights $\alpha_1, ..., \alpha_N$ that sum to 1. The final representation $\hat{x}$ is the weighted average of the inputs.

$$s_i = align(q, x_i) \tag{2.30}$$

$$\alpha_i = softmax(s_i) = \frac{e^{s_i}}{\sum_j e^{s_j}} \tag{2.31}$$

$$\hat{x} = \sum_i \alpha_i x_i \tag{2.32}$$

It turns out that the alignment score function is of high importance, a collection of popular alignment score functions follows.

General Attention

$$align(q, x_i) = s^\tau W_\alpha x_i \tag{2.33}$$

Additive Attention

$$align(q, x_i) = v_\alpha^\tau \tanh(W_{\alpha[s;x_i]}) \tag{2.34}$$

Dot-Product Attention

$$align(q, x_i) = s^\tau x_i \tag{2.35}$$

Scaled Dot-Product Attention

$$align(q, x_i) = \frac{s^\tau x_i}{\sqrt{N}} \tag{2.36}$$

2.6 Transformers

The Transformer is a deep learning model introduced by Vaswani et al. [92] used primarily in Natural Language Processing. The Transformer is based in the Attention Mechanism idea, discussed in subsection 2.5.6. Like RNNs, Transformers are designed to handle sequential data, however do not require that the sequential data be processed in order. Since their introduction, Transformers have become the model of choice for tackling many problems in NLP. As Transformers allows for much more parallelization than RNNs during training, it has enabled training on larger datasets and led to the development of pretrained systems such as BERT (Bidirectional Encoder Representations from Transformers) [24] and GPT (Generative Pre-trained Transformer) [19], which have been trained with huge general language datasets, and can be fine-tuned to specific language tasks.

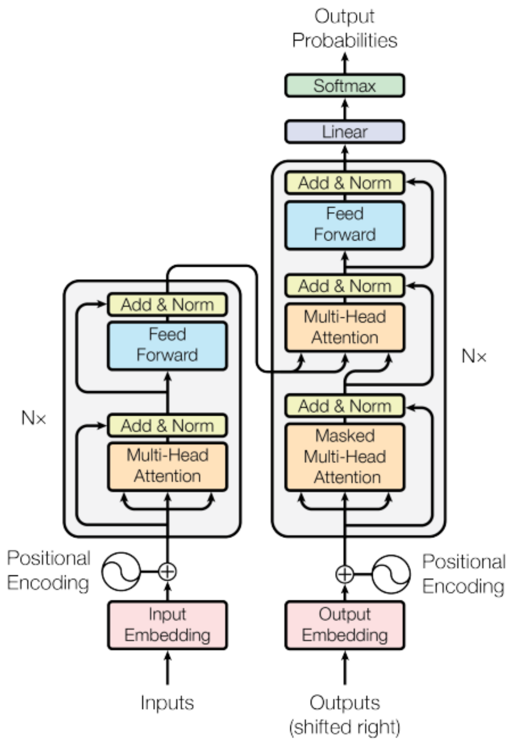


Figure 2.8: The Transformer architecture. From [92]

The Transformer is an encoder decoder architecture. he encoder consists of a set of encoding layers that processes the input iteratively one layer after another and the decoder consists of a set of decoding layers that does the same thing to the output of the encoder. Encoders and decoders, consists of identical subsystems, that have different parameter values. Encoders and decoders accept inputs from the lower layers and carry them to the higher ones. Each encoder consists of two subsystems. The first is a self-attention layer, that allows the encoder to hold the dependencies that the input under study may have with the other inputs in the sequence. The second is a feed forward neural network for additional processing of the outputs, and residual connections and layer normalization units. Similar is the decoders architecture with an addition of a intermediate encoder-decoder attention mechanism. This subsystem is responsible to identify and focus on specific elements of the input that has already been encoded by the encoder.

The feed forward neural network does not retain the correlations between the input units. The system which is responsible for the correlations between the input units is the self-attention. The calculation of the outputs of the Self-Attention system is essentially done in two basic steps. The first step is to create three vectors from the input. Specifically, Query, Key and Value vectors need to be created. This vectors resulting from the multiplication of inputs with specific arrays, the parameters of which are optimized during the training process. Given the three vectors that make up the self-attention mechanism, Query (Q), Key (K) and Value (V), we can define self-attention by Equation (2.37), where $\frac{1}{\sqrt{d_k}}$ is a scaling factor, dependant on the dimension of vector K.

$$Attention(Q,K,V) = softmax(\frac{QK^\intercal}{sqrtd_k})V \tag{2.37}$$

One set of W_Q, W_K, W_V matrices is called an attention head.In practice multi-head attention is used. In the original Transformer paper, the architecture has 8 attention heads, that make 8 different sets of matrices W_Q, W_K and W_V for each self-attention subsystem for each encoder and decoder. Different heads can give a different form of attention that the model will focus. The output of the attention mechanism is multiplied by another learnable matrix W_O before feed in the feed forward neural network subsystem.

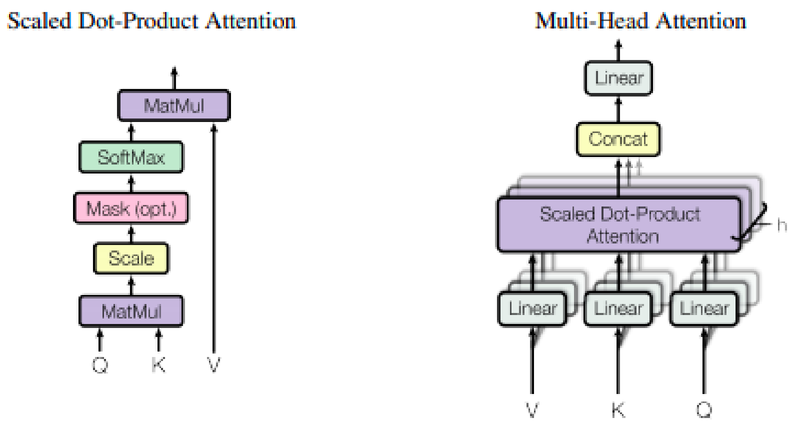


Figure 2.9: Left: Scaled Dot-Product Attention. Right: Multi-Head Attention consists of several attention layers in parallel. From [92]

Another issue to be addressed is the position of inputs. Some vectors need to be introduced which will give the system a sense of order in the input sequence. This vectors are known as positional embeddings. Those embeddings are added to the corresponding input embeddings, and converts the input to vectors with internal representation of time characteristics. The first encoder takes positional information and embeddings of the input sequence as its input, rather than encodings.

Transformers typically undergo semi-supervised learning involving unsupervised pretraining followed by supervised fine-tuning. Pretraining is typically done on a much larger dataset than fine-tuning, due to the restricted availability of labeled training data.

2.7 Evaluation of Performance

A learning algorithm fulfills its purpose when produces hypothesis that performs well on predicting examples that have never been seen before. Obviously, a prediction is good when it turns out to be true. Therefore we can evaluate the quality of a hypothesis by checking the accordance of the predictions with the ground truth, when the latter becomes known. We can achieve so by keeping a separate set of known data, that is named test set. If we train in all available data, we should collect some more data for testing, therefore is more practical to adopt the following methodology:

- Select a large dataset.
- Split the dataset in two separate sets, a training set and a test set.

- Apply the learning algorithm on the train set and create an hypothesis h.
- Get predictions of h on test set and compute the performance metrics.
- Repeat steps 2-4 for different sizes of training sets and different random test sets.

learning curve The result of the above procedure is a set of performance metrics, that can be processed to draw the mean performance as a function of the training set size, that is known as the learning curve. As training set size augments, learning curves that are improving are a good sign that there is a pattern on the data and the learning algorithm recognizes that pattern.

peeking We have to note that the learning algorithm must not be aware of the test set, before the testing phase. This principle is easy to violate as follows: a learning algorithm may have some levers that regulate its performance, we create various hypotheses for different settings of that levers, compute performance metrics on a test set and choose the hypothesis with the best performance on the test set. Unfortunately we selected a hypothesis in a way that information of the test set leaked and became known to the learning algorithm. This common misbehavior is called peeking. A solution is to use a new test set to evaluate the performance of the selected hypothesis.

Cross-validation Cross-validation is a technique to reduce overfitting (see also section section 2.8), that can be applied on all learning algorithms. The key concept is to compute evaluation metrics of a hypothesis over unseen data. That can be achieved by keeping a subset of data aside and using it for validating the performance of a hypothesis concluded from the rest data. In K-fold cross-validation we conduct K experiments, keeping each time as a validation set a different $1/K$ part of data, and extract a mean metric. Popular values for K is 5 and 10.

2.8 Overfitting and Regularization

2.8.1 Overfitting and Underfitting

When there is a large set of possible cases, we must be careful not to use the resulting freedom to find meaningless normality in the data. This problem is called overfitting. Therefore overfitting is the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably. Overfitting which is a very general phenomenon, occurs even when the target function is not random and affects all types of learning algorithms.

Underfitting, on the opposite, occurs when a model cannot sufficiently capture the structure of the data, or the model does not have the capacity to fully learn the data. In an underfitted model, some parameters that would appear in a correctly trained model are missing and such a model will have poor predictive performance. An extreme example of underfitting is fitting a linear model to non-linear data. On Figure 2.10 we can observe undrefitting and overfitting compared to an ideal fit.

2.8.2 Regularization

In order to succeed the ideal fit, face overfitting and improve the generalization of a model to unseen data, various techniques were developed. Regularization refers to a set of techniques used to prevent overfitting by adding information. Most regularization strategies aim at balancing the bias-variance trade-off. That means achieving a lower generalization error at the expense of increasing training error. One way to achieve so is by regulating the model capacity, though determining a fix number of model parameters does not secures a good fit. The best strategy is to consider complex function families and use regularization techniques to control complexity.

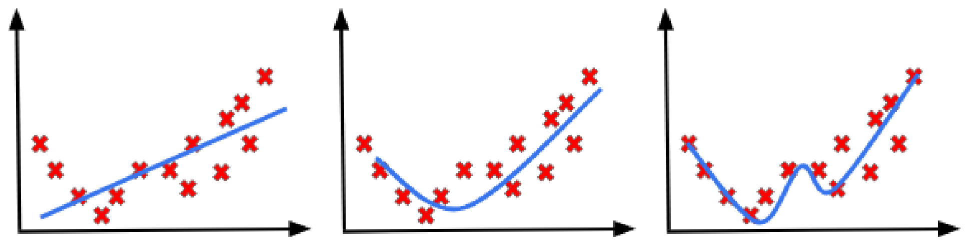


Figure 2.10: From left to right: Underfitting, the model does not have capacity to fully learn the data. Ideal fit, the model learns the underlying data structure. Overfitting, the model is too complex and learns noise over the training data.

Parameter Norm Penalties A common approach to limit the complexity of a model is to penalize the norm of the model parameters. This is achieved by adding to the loss function $J(w)$ a regularization term $\Omega(w)$, multiplied by a constant $\lambda \in [0, 1]$ that controls the amplitude of the regularization. The overall loss will be:

$$J_{new}(w) = \frac{1}{n}\sum_{i=0}^{n} \mathcal{L}(f(x^i; w), y^{(i)}) + \lambda\Omega(w) \tag{2.38}$$

Two commonly used penalties are the following:

L_1 regularization

$$\Omega(w) = \|w\|_1 = \sum_{i=1} |w_i| \tag{2.39}$$

The update weights rule of gradient descent algorithm will then be

$$w \leftarrow w - \gamma(\frac{\partial J(w)}{\partial w} + \lambda sign(w)) \tag{2.40}$$

L_1 regularization leads the model to be sparse. At each update, a constant factor is subtracted from all weights with a sign equal to $sign(w)$. As this can drive some of the model wights to zero, L_1 regularization acts as a feature selection mechanism.

L_2 regularization, also known as weight decay.

$$\Omega(w) = \frac{1}{2}\|w\|_2^2 \tag{2.41}$$

The update weights rule of gradient descent algorithm will then be

$$w \leftarrow w - \gamma(\frac{\partial J(w)}{\partial w} + \lambda w) \tag{2.42}$$

At each update, a term proportional to their magnitude is subtracted from all weights. This causes the weights to decay over time, so the technique is named weight decay. As large weights are penalized more than small ones, L_2 regularization encourages the weights to remain small.

Sometimes L_1 and L_2 regularizations are combined, in a technique known as elastic net regularization.

$$\Omega(w) = (1 - \lambda)\|w\|_1 + \frac{\lambda}{2}\|w\|_2^2 \tag{2.43}$$

Dropout Dropout is a simple yet effective, frequently used regularization method for neural networks. Given a probability $p > 0$, during training each unit is dropped with that probability. Being dropped means that the unit is ignored during both forward and backward pass and their activations are set to zero. In Figure 2.11 we can observe the effect of dropout on a neural network during training. At test time, all units are employed, but they are scaled by a factor $1/p$ to account for the missing activations during training. Dropout forces network to not rely on any node and prevents units from forming co-dependencies amongst each other. A usual value for p is 0.5.

Given a model with N units, each of which can be dropped. Training that neural network with dropout can be seen as training a collection of 2^N subnetworks that share some of their parameters. Therefore dropout is a form of ensembling an exponential number of models by training and evaluating just one model.

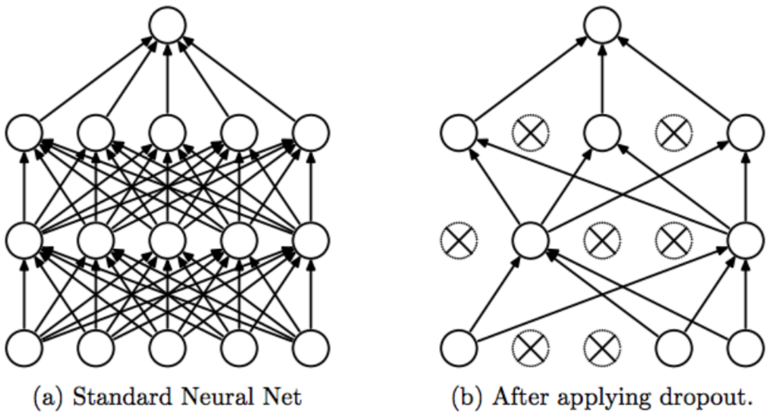


Figure 2.11: The effect of dropout during training.

Data Augmentation A simple way to reduce the risk of overfitting is to use more training data. Given the that is not possible to primarily collect more data, one ca create synthetic data from the available dataset. This method is called data augmentation, and the way to create such data points depends on the specific domain and type of data. Regarding images it is common to use small transforms such as flipping, rotating, re-scaling and cropping. For natural language data augmentation is less common, yet it can be done by synonym replacement. In general, data augmentation can also be achieved by adding small random noise to the data in order to improve the model robustness.

Early stopping When an iterative training method is used, a common regularisation technique is early stopping. As the training progresses, a model tends to fit better the training data. Up to a point, this also improves the model performance on data outside of the training set. After that point, however, that comes at the cost of overfitting, as the generalization error increases and the model learn the noise in the training data. Early stopping rules provide general guidance on the number of iterations to run, in order to avoid overfit. In practise we monitor at each epoch the performance of the model on a validation set. Validation set is a part of the original training set, that each time is kept out of the training process. The error on the validation set is used as a proxy for the generalization error. Early stopping terminates the training as soon as the validation loss stops decreasing. A visualization of training and validation errors evolution over time can be observed on Figure 2.12. Sometimes the validation loss may start decreasing again after a few epochs, so it is common to allow the model to continue training for a predefined number of epochs, known as training patience. In essence, early stopping is tuning the hyperparameter number of epochs.

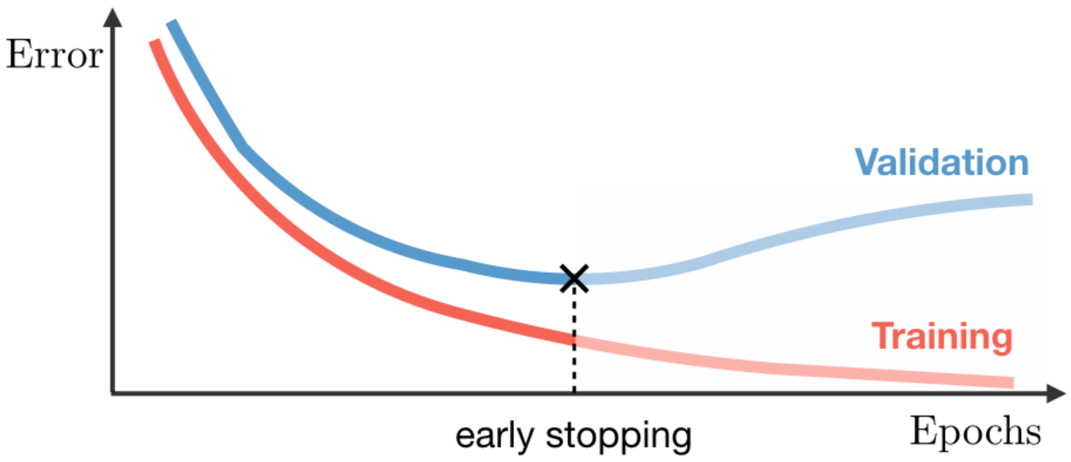


Figure 2.12: Training and validation error over time. Early stopping prevents the increase of generalization error as measured on validation set.

2.9 Transfer Learning

In the classic supervised learning scenario of machine learning, if we aim to train a model for a specific task and domain, we assume that we have sufficient labeled data for the same task and domain. Typically a model is trained on this dataset and is expected to perform well on unseen data of the same task and domain. It is expected the data to be i.i.d. (independent and identically distributed random variables). If the task or domain changes, then new labelled data of the same task or domain are needed in order to train a new domain specific model.

The above traditional learning paradigm breaks down when we do not have sufficient labelled data for the desired task or domain to train a specific model. Transfer learning faces that limitation by leveraging data of other task and domains, known as the source task and source domain. The knowledge gained in solving the source task in the source domain is used to solve the target task in the target domain.

Transfer learning refer to the situation where what has been learned in one setting is exploited to improve generalization in another setting. In transfer learning, the learner must perform two or more different tasks, but we assume that many of the factors that explain the variations in the first task are relevant to the variations that need to be captured for learning the second task. In the sub-case of transductive transfer learning, also known as domain adaptation, the task (and the optimal input-to-output mapping) remains the same between each setting, but the input distribution is slightly different. We further discuss domain adaptation in Chapter 4.

A domain D consists of a feature space X and a marginal probability distribution $P(x)$ over the feature space, where $x \in X$. Given a domain $D = \{X, P(X)\}$, a task T consists of a label space Y and a conditional probability distribution $P(Y|X)$ that is typically learned from the training data $\{x_i, y_i\}$ with $x_i \in X$ and $y_i \in Y$. Given a source domain D_S, a corresponding source task T_S, as well as a target domain D_T and a target task T_T, transfer learning's objective is to enable us to learn the target conditional probability distribution $P(Y_T|X_T)$ in D_T with the information gained from D_S and T_S.

Transfer learning improve performance of models in three ways. First is the initial performance achievable in the target task using only the transferred knowledge, before any further learning is done, compared to the initial performance of an ignorant agent. Second is the amount of time it takes to fully learn the target task given the transferred knowledge compared to the amount of time to learn it from scratch. Third is the final performance level achievable in the target task compared to the final level without transfer.

A special case of transfer learning involves a scenario where source task is unsupervised and target task is supervised. This is particularly interesting because we often have large amounts of unlabeled training data, yet relatively little labeled training data. Training with supervised techniques on the labeled subset often results in overfitting. By learning good representations from the unlabeled data, we can perform better in the supervised learning task. This transfer learning case is called unsupervised pretraining. This procedure is an example of how a representation learned for one task (unsupervised learning, trying to capture the shape of the input distribution) can sometimes be useful for another task (supervised learning). It is called pretraining, because it is supposed to be only a first step before a joint training algorithm is applied to fine-tune all the layers together.

Chapter 3

Natural Language Processing

3.1 Introduction

Natural Language Processing (NLP) is a subfield of computer science, artificial intelligence and linguistics concerned with the interactions between computers and human languages. NLP faces the the computer understanding, analysis, manipulation and generation of language, in written and spoken form. The goal of NLP is to identify the algorithms and techniques needed for a computer to exhibit various forms of linguistic behavior. That is to design, implement and test systems that process natural languages for practical applications. The levels of language that NLP examines are:

Phonology In this first level, the speech sounds of languages are examined.

Morphology This level deals with the componential nature of words, which are composed of morphemes, the smallest units of meaning.

Lexical At lexical level, systems interpret the meaning of individual words.

Syntactic Syntactic level focuses on analyzing the words in a sentence in order to uncover the grammatical structure of the sentence.

Semantic Semantic processing determines the meaning of a sentence. This level of processing includes the semantic disambiguation of the corresponding words, that may have multiple senses.

Discourse The discourse level of NLP works with units of text longer than a sentence. Discourse focuses on the properties of the text as a whole that convey meaning by making connections between component sentences, and does not interpret text as just concatenated sentences.

Pragmatic This level is concerned with the purposeful use of language in situations and utilizes context over and above the contents of the text for understanding. The goal is to explain how extra meaning is read into texts without actually being encoded in them.

3.2 NLP Tasks

NLP includes a large list of tasks, that are either directly connected to real world applications or serve as subtasks to aid in solving other tasks. In this section we will shortly present a list of common tasks and then we will further present sentiment analysis task, which mainly concerns this thesis.

3.2.1 Common Tasks

Tokenization Also known as word segmentation, tokenization is the process of seperating a chunk of continuous text into separate words.

Part-of-speech tagging Given a sentence, determine the part of speech (POS) for each word. Many words, especially common ones, can serve as multiple parts of speech.

Parsing Determine the parse tree (grammatical analysis) of a given sentence. The grammar for natural languages is ambiguous and typical sentences have multiple possible analyses: perhaps surprisingly, for a typical sentence there may be thousands of potential parses (most of which will seem completely nonsensical to a human)

Named entity recognition Given a piece of text, determine which items in the text map to proper names, such as people or places, and what the type of each such name is (e.g. person, location, organization).

Topic segmentation Given a chunk of text, separate it into segments each of which is devoted to a topic, and identify the topic of the segment.

Text summarization The process of shortening a set of data computationally, to create a subset (a summary) that represents the most important or relevant information within the original content.

Machine translation Automatically translate text from one human language to another.

Natural language generation Convert information from computer databases or semantic intents into readable human language.

Natural language understanding Convert chunks of text into more formal representations such as first-order logic structures that are easier for computer programs to manipulate.

Question answering Given a human-language question, determine its answer.

3.2.2 Sentiment analysis

Sentiment analysis, also called opinion mining or emotion AI, is the field of study that analyzes people's opinions, sentiments, evaluations, appraisals, attitudes and emotions towards entities such as products, services, organizations, individuals, issues, events, topics and their attributes. Sentiment analysis deals with extracting subjective information usually from a set of documents, often using online reviews to determine ”polarity” about specific objects. It is especially useful for identifying trends of public opinion in social media, for marketing. Sentiment analysis in its simplest form, aims to detect positive or negative feelings from text.

3.3 Word vectors-Embeddings

An essential step in Natural Language Processing is to represent the text, and especially its components such as words, in a suitable form to be used as input in the various machine learning models. The aim is to create vector representations, one for each word, that provide a sense of similarity or dissimilarity between words that are indeed similar or irrelevant. These vector representations are known as word vectors or word embeddings. Word embeddings is, therefore, a collective name for a set of language modeling and feature learning techniques where linguistic entities, words or phrases, are mapped to vectors of real numbers. Conceptually, embedding, involve a mathematical embedding from a high dimensionality space to a continuous vector space with a much lower dimension.

3.3.1 Denotional approach

A semantic approach to word vectors, known as denotional semantics, considers words as unique symbols and creates sparse representations, from which is difficult to draw a conclusion about similarity between words. Such methods are words as vocabulary indices and one-hot vectors.

3.3.2 Vocabulary indices

The first and most naif, denotional method is to use the indices that correspond to the word in the vocabulary. Lets consider an example vocabulary V : $V[0] =$ "$abate$", $V[1] =$ "$abdicate$", $V[2] =$ "$aberrant$", $V[3] =$ "$reduce$", $V[4] =$ "$resign$". The word "$abate$" is mapped to the number $i_{abate} = 0$ and the word "$reduce$" to the number $i_{reduce} = 3$. The similarity of these two words has not been captured in their indices, and no semantic relation can be extracted from the arithmetic relation of appointed indices.

3.3.3 One-hot vectors

A one-hot vector is a group of bits among which only a single bit is 1 and all the others are 0. Therefore, a one-hot word vector is a $1 \times |V|$ vector used to distinguish each word in the vocabulary from every other. Each vector consists of zeros in all cells with the exception of a single one in a cell that identifies the specific word. A relative advantage of the method when compared with vocabulary indices, is that representations does not have a meaningless relation among them, but in fact they have no relation at all. Lets consider again the example vocabulary $V = \{abate, abdicate, aberrant, reduce, resign\}$. We will have the following one-hot vectors: $w_{abate} = \begin{bmatrix}1 & 0 & 0 & 0 & 0\end{bmatrix}$, $w_{abdicate} = \begin{bmatrix}0 & 1 & 0 & 0 & 0\end{bmatrix}$, $w_{aberrant} = \begin{bmatrix}0 & 0 & 1 & 0 & 0\end{bmatrix}$, $w_{reduce} = \begin{bmatrix}0 & 0 & 0 & 1 & 0\end{bmatrix}$ and $w_{resign} = \begin{bmatrix}0 & 0 & 0 & 0 & 1\end{bmatrix}$. The corresponding matrix that describes the method will be of $|V| \times |V|$ dimension and in the following form:

$$\begin{bmatrix} w_{abate} \\ w_{abdicate} \\ w_{aberrant} \\ w_{reduce} \\ w_{resign} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

The above matrix is of extreme sparsity, as most of information in it is just zeros. Given also that the vocabulary size in practice will be big enough to express the language, a memory issue occurs. The word vectors does not have any dependency on each other, that means no semantic or relational information can be extracted. The vectors are, in fact, linearly independent, as shown for example in Equation (3.2).

$$w_{abate}^{T} w_{abdicate} = w_{abdicate}^{T} w_{resign} = 0 \tag{3.2}$$

3.3.4 Distributional hypothesis

In contrast to denotional semantics, distributional semantics, based in the distributional hypothesis, create representations based in the context in which a word is found in a corpus, and therefore succeeds both dimensionality reduction and giving a sense of similarity among similar words. The idea of the distributional hypothesis is that the distribution of words in a text holds a relationship with their corresponding meanings. More specifically, the more semantically similar two words are, the more they will tend to show up in similar contexts and with similar distributions. The hypothesis is also known as "words that occur in the same contexts tend to have similar meanings", and as "a word is characterized by the company it keeps". Based in the hypothesis, distributional semantics attempt to capture meanings of linguistic entities from their usage in language. That is two words that are considered to be semantically similar are expected to occur in similar contexts, and words that occur in similar contexts should be considered similar.

3.3.5 Word2Vec

Word2Vec [61] is a shallow, two-layer neural network which is trained to reconstruct linguistic contexts of words. Given an input of large corpus of words, it produces a vector space, typically of some hundred dimensions, with each word in the corpus being assigned a corresponding vector in the space. Word vectors are located in the space such that words that share common contexts in the corpus are located in close proximity to one another in the space. Word2Vec has two forms, the continuous bag of words (CBOW) model and the Skip-Gram model. When the feature vector assigned to a word cannot be used to accurately predict that word's context, the components of the vector are adjusted. The vectors of words judged similar by their context are nudged closer together by adjusting the numbers in the vector.

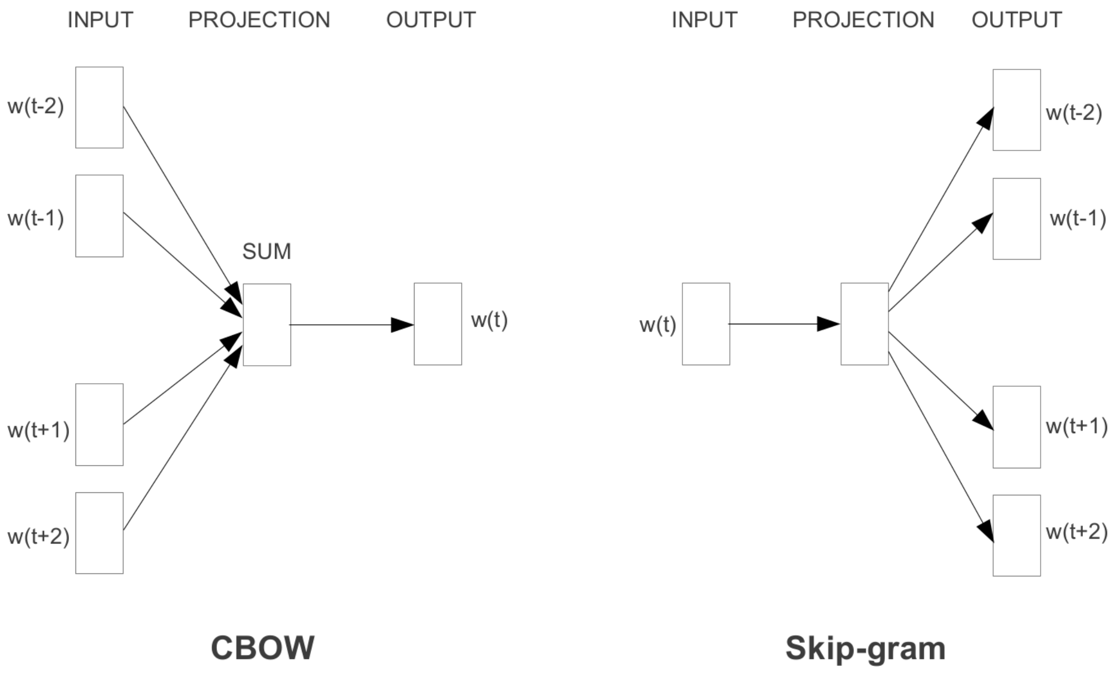


Figure 3.1: Word2Vec training modes. Figure from [61]

The continuous bag of words ($CBOW$) model is trained to learn word embeddings by predicting a specific word given its neighbors. Supposing we want to predict a word w_i, the input to the model could be w_{i-2}, w_{i-1}, w_{i+1} and w_{i+2}, the preceding and following words of the target. $Skip-Gram$ is the exact opposite of $CBOW$, as now the central word w_i is given as input and the output would be w_{i-2}, w_{i-1}, w_{i+1} and w_{i+2}. The Skip-Gram task is therefore to learn word emebeddings by training a model to predict context given a word. The two models are illustrated in Figure 3.1.

Given enough data, usage and contexts, Word2vec can make highly accurate guesses about a word's meaning based on past appearances. Those guesses can be used to establish a word's association with other words (e.g. "king" is to "man" what "queen" is to "woman"). It becomes apparent that the vectors capture some general, and in fact quite useful, semantic information about words and their relationships to one another, e.g. male-female, verb tense and even country-capital relationships between words.

3.3.6 GloVe

GloVe [68] is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. GloVe is essentially a log-bilinear model with a weighted least-squares objective. The training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence. The main intuition underlying the model is the observation that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning. The GloVe model is trained on the non-zero entries of a global word-word co-occurrence matrix, which tabulates how frequently words co-occur with one another in a given corpus. Populating this matrix requires a single pass through the entire corpus to collect the statistics.

3.3.7 Contextual Word Embeddings

The distributional approaches aggregate the contexts in which a term occurs in a corpus. The result is a context-free, or else context-independent word representation. Word embeddings described so far are static. Nevertheless, usually a particular word can be used in different sentences with a completely different meaning. Out of context, each word has multiple meanings.An obvious problem that occurs is that polysemous words (words with obvious multiple senses) cannot be modeled properly. For example, in the following sentences: "I dream of surfing the perfect wave." and "Will there be another wave of illness in the fall?", word "wave" has quite different meaning. So static representations for words are quite insufficient solutions for understanding text. Therefore, it is proposed to represent words, depending on the context each time found. A famous algorithm for contextualized word representation is ELMo (Embeddings from Language Models) [69], that will presented in Section 3.5.

3.4 Language Modeling

Language Models (LMs) compute the probability distribution over a sequence of words. Assigning probabilities to sentences is a task of great importance, used in various applications, such as speech recognition or spell correction, and in general when the most likely sequence is needed. The probability of a sequence of K words $w_1, w_2, ..., w_K$ is denoted as $P(w_1, w_2, ..., w_K)$. Given the definition of conditional probabilities, the general chain rule and by applying it in order to compute the joint probability of words in a sentence we get the Equation (3.3) that describes language modeling.

$$P(w_1, w_2, ..., w_K) = \prod_{i=1}^{K} P(w_i|w_1, ..., w_{i-1}) \tag{3.3}$$

Since the number of words coming before a word w_i varies, depending on its location in the input, $P(w_1, w_2, ..., w_K)$ is usually conditioned on an window of n previous words rather than all previous words.

$$P(w_1, w_2, ..., w_K) = \prod_{i=1}^{K} P(w_i|w_1, ..., w_{i-1}) \approx \prod_{i=1}^{K} P(w_i|w_{i-n}, ..., w_{i-1}) \tag{3.4}$$

Equation (3.4) is used in various applications, such as machine translation and speech recognition, in order to determine a word sequence that is an accurate interpretation, transcription or translation of an input sentence.

3.4.1 n-gram Language Models

In order to obtain the probabilities described by Equation (3.4), we should count and compare the respective n-grams and word frequencies. This is called an n-gram language model. For instance, if we consider a bi-gram, the frequency of each bi-gram would be divided by the frequency of the corresponding uni-gram of the second term. Equations (3.5) and (3.6) give the respective relations for bi-gram and tri-gram models.

$$p(w_2|w_1) = \frac{count(w1, w2)}{count(w1)} \tag{3.5}$$

$$p(w_3|w_2, w_1) = \frac{count(w1, w2, w3)}{count(w1, w2)} \tag{3.6}$$

Those models focus on making predictions based on a fixed window of context, the n) previous words, used to predict the next word. The selection of an appropriate window length for the context is an important question to be answered. In some cases the window of past n words may not be sufficient to capture the context. For example lets consider the sentence "I wanted to borrow a book and so I went to the [MASKED]". One would reasonably guess that the masked word is "library". However, even a 4-gram language model would try to predict the hidden word from the previous three words "went to the". It is obvious that this model ignores the real context of the sentence. There are two main problems with n-gram language models that occur, sparsity and storage.

Sparsity problems arise due to two issues. Firstly, given the numerator of Equation (3.6), it possible the tri-gram w_1, w_2, w_3 never appear together in the corpus and so the probability will be 0. To solve the issue it is usual to to add a small δ for each word count. This technique is known as smoothing. Secondly, a problem occurs also with the denominator of Equation (3.6) If w_1, w_2 never occurred as a bi-gram in the corpus, then the probability cannot be calculated. To solve this a common solution, called backoff, is to condition on w_2 alone. Increasing n makes sparsity problems worse.

Storage problems also arise with n-gram language models. We need to store the count for all n-grams we see in a corpus. As n increases or the corpus size increases or the vocabulary used increases, the model size increases as well. To overcome both problems, typically is selected $n \leq 5$.

3.4.2 Neural Language Models

To address the shortcomings of traditional language models, non-linear neural network models have been proposed. That models allow conditioning on increasingly large context sizes with a linear increase in the number of parameters. Such a neural probabilistic language model was proposed by Bengio et al. [13]. This model takes as input vector representations of a word window of n previous words, which are looked up in a table C. These vectors are in fact word embeddings. These word embeddings are then concatenated and given as input to hidden layer, whose output provided to a softmax layer (Figure 3.2).

$$\begin{aligned} x &= [C(w_1); C(w_2); ...; C(w_n)] \\ \hat{y} = P(w_i|w_{1:k}) = LM(w_{1:k}) &= softmax(hW_2 + b_2) \\ h &= g(xW_1 + b_1) \\ C(w) &= E_{[w]} \end{aligned} \tag{3.7}$$

where $w_i \in V, E \in \mathbb{R}^{|V| \times d_w}, W_1 \in \mathbb{R}^{nd_w \times d_h}, b_1 \in \mathbb{R}^{d_h}, W_2 \in \mathbb{R}^{d_h \times |V|}, b_2 \in \mathbb{R}^{|V|}$. V is a finite vocabulary.

3.4.3 Perplexity

There are several metrics for evaluating language modeling. A natural evaluation of language models is using perplexity over unseen sentences. Perplexity is an information theoretic measurement of

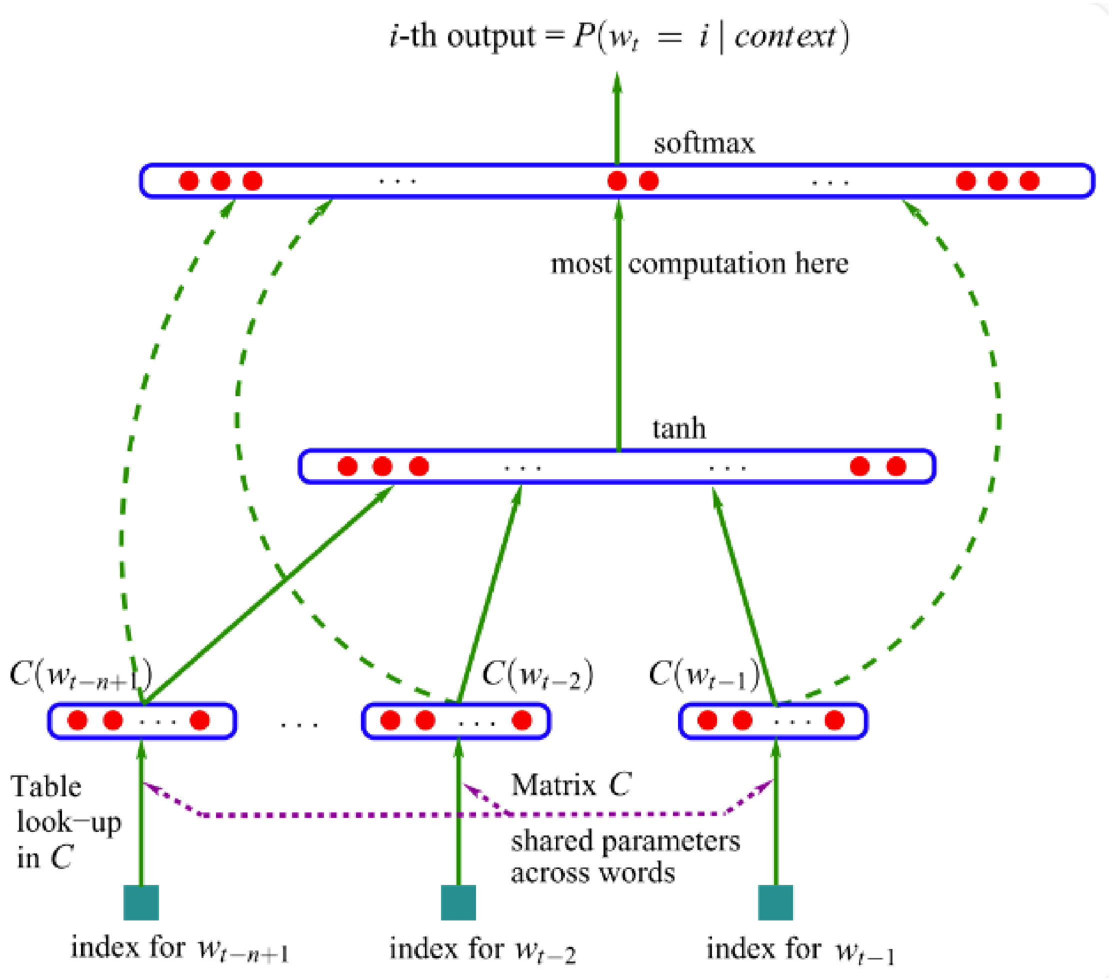


Figure 3.2: The first deep neural network architecture model presented in [13]

how well a probability model predicts a sample. Low perplexity values indicate a better fit. Given a text corpus of n words $w_1, w_2, ..., w_n$ and a language model LM assigning a probability to a word based on its history, the perplexity of a LM with respect to the corpus is:

$$Perplexity = 2^{-\frac{1}{2}\sum_{i=1}^{n} log_2 LM(w_i|w_{1:i-1})} \tag{3.8}$$

Good language models, that indeed reflect of real language usage, will assign high probabilities to the events in the corpus, resulting lower perplexity values. Therefore, perplexity measure is a good indicator of the quality of a language model. Perplexities are corpus specific, so perplexities of two language models are only comparable with respect to the same evaluation corpus.

3.5 ELMo

$ELMo$ embeddings [69] are deep contextualized word representations that offer high quality representations for language, modeling both complex characteristics of word use and adjusting them in different linguistic contexts. The vectors are derived from a bi-directional LSTM that is trained with a coupled language model (LM) objective on a large text corpus. ELMo representations are a function of all the internal layers of the bi-directional language model. However, the weighting of the ELMo embeddings needs to be carefully tuned for every different task. Therefore, the proposed method, although very effective, has some limitations and requires task-specific architectures.

3.6 BERT

Bidirectional Encoder Representations from Transformers (BERT) [24] is a pre-trained model of deep bidirectional transformers for language understanding, which is then fine-tuned for a task. BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be finetuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task specific architecture modifications. During pre-training, the model is trained on unlabeled data over different pre-training tasks. For BERT pre-training two unsupervised tasks are used, Masked Language Modeling (MLM) and Next Sentence Prediction. For finetuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters.

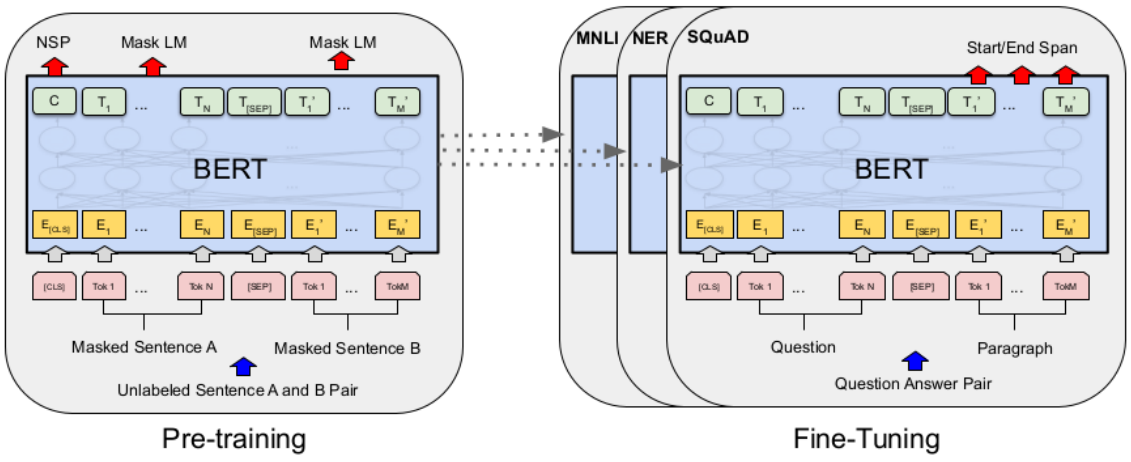


Figure 3.3: Pre-training and fine-tuning procedures for BERT. From [24]

BERT's model architecture is a multi-layer bidirectional Transformer encoder. The original English-language BERT model comes with two pre-trained general types: (1) the $BERT_{BASE}$ model, a 12-layer, 768-hidden, 12-heads, 110M parameter neural network architecture, and (2) the $BERT_{LARGE}$ model, a 24-layer, 1024-hidden, 16-heads, 340M parameter neural network architecture; both of which were trained on the BooksCorpus with 800M words, and a version of the English Wikipedia with 2,500M words.

BERT input representation is able to unambiguously represent both a single sentence and a pair of sentences (e.g. Question, Answer) in one token sequence. WordPiece embeddings [] with a 30,000 token vocabulary are used. The first token of every sequence is always a special classification token ([CLS]). The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks. Sentence pairs are packed together into a single sequence. Sentences are differentiated in two ways, a special token is used ($[SEP]$) and d a learned embedding to every token indicating whether it belongs to sentence A or sentence B is added. For a given token, its input representation is constructed by summing the corresponding token, segment, and position embeddings.

Intuitively, it is reasonable to believe that a deep bidirectional model is strictly more powerful than either a left-to-right model or the shallow concatenation of a left-to-right and a right-to-left model. Unfortunately, bidirectional conditioning would allow each word to indirectly "see itself", and the model could trivially predict the target word in a multi-layered context. In order to train a deep bidirectional representation, Masked Language Modeling is proposed, by simply mask some percentage of the input tokens at random. In this case, the final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary, as in a standard LM. In BERT 15% of all WordPiece tokens in each sequence are masked at random. As masking creates a mismatch between pre-training and fine-tuning masked words are not always replaced by a mask token, but at 10% chance remains unchanged and at 10% chance is changed by a random token.

The relationship between sentences, is of high importance in some NLP tasks. In order to train a model that understands sentence relationships, BERT pre-train for a binarized next sentence prediction task that can be trivially generated from any monolingual corpus. Specifically, when choosing the sentences A and B for each pretraining example, 50% of the time B is the actual next sentence that follows A (labeled as $IsNext$), and 50% of the time it is a random sentence from the corpus (labeled as $NotNext$).

Fine-tuning is straightforward since the self-attention mechanism in the Transformer allows BERT to model many downstream tasks, whether they involve single text or text pairs, by swapping out the appropriate inputs and outputs. At the input, sentence A and sentence B from pre-training are analogous to (1) sentence pairs in paraphrasing, (2) hypothesis-premise pairs in entailment, (3) question-passage pairs in question answering, and (4) a degenerate text - $\emptyset$ pair in text classification or sequence tagging. At the output, the token representations are fed into an output layer for token level tasks, such as sequence tagging or question answering, and the $[CLS]$ representation is fed into an output layer for classification, such as entailment or sentiment analysis.

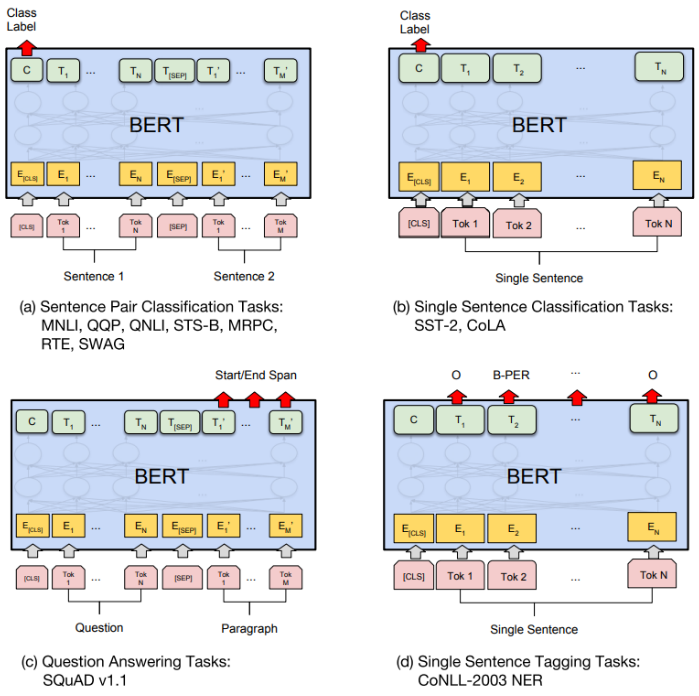

Class Label
BERT
Sentence 1
Sentence 2
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG
Class Label
BERT
Single Sentence
(b) Single Sentence Classification Tasks: SST-2, CoLA
Start/End Span
BERT
Question
Paragraph
(c) Question Answering Tasks: SQuAD v1.1
O
B-PER
O
BERT
Single Sentence
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Figure 3.4: BERT applied in various tasks. From [24]

Chapter 4

Unsupervised Domain Adaptation in NLP

4.1 Introduction

Domain adaptation is a field of machine learning, a specific sub-case of transfer learning, that arises when we aim at learning from a source data distribution a well performing model on a different, but related, target data distribution. As shown in Figure 4.1, domain adaptation, also called transductive transfer learning, takes place when source and target marginal distributions are different, but the task we seek to learn is the same on the two domains. Domain adaptation is the ability to apply an algorithm trained in one or more source domains to a different, but related, target domain. In domain adaptation, the source and target domains all have the same feature space, but different distributions. A domain shift also occurs at deployment of machine learning systems, as usually there is a change in the data distribution between the training dataset and real data.

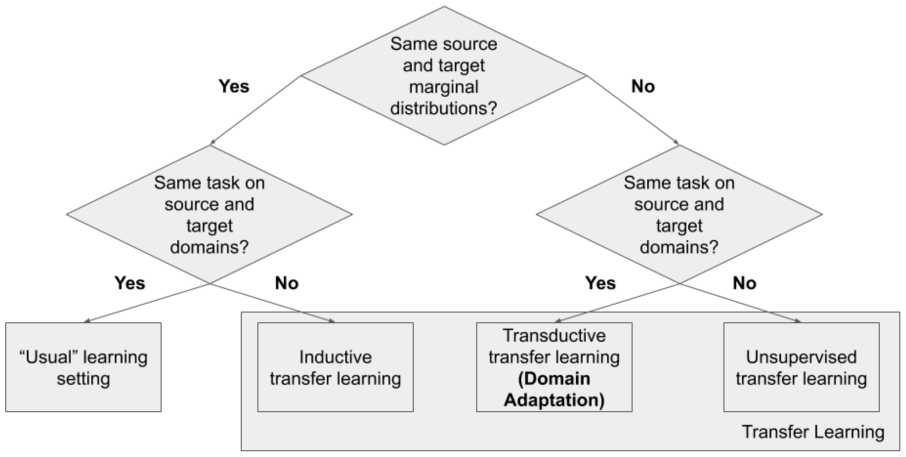

Same source and target marginal distributions?
Yes
No
Same task on source and target domains?
Yes
No
Same task on source and target domains?
Yes
No
"Usual" learning setting
Inductive transfer learning
Transductive transfer learning **(Domain Adaptation)**
Unsupervised transfer learning
Transfer Learning

Figure 4.1: Distinction between usual machine learning setting and transfer learning, and positioning of domain adaptation.

The lack of portability of NLP models to new conditions remains a central issue in NLP. For many target applications, labeled data is lacking, and even for pretraining general models data might be scarce. In practice annotation is a substantial time-requiring and costly manual effort and is not easily scalable to new application targets. Unsupervised domain adaptation mitigates the domain shift issue by learning only from unlabeled target data, which is typically available for both source and target domains. Unsupervised domain adaptation fits the classical real-world scenario better, in which labeled data in the target domain is absent, but unlabeled data might be abundant. Therefore it is an elegant and scalable solution.

4.2 NLP Tasks for Unsupervised Domain Adaptation

The most studied NLP task under the unsupervised domain adaptation scenario is sentiment analysis. Most works verify their results in the well studied benchmark of multi domain Amazon reviews [14]. Other notable tasks used are mostly binary classification and structural prediction tasks. Related tasks include but are not limited to relation extraction, name entity recognition, dependency parsing, part-of-speech tagging, natural language inference, duplicate question detection and machine reading.

4.3 Notation – Problem Setting

In general, in domain adaptation we consider classification tasks where X is the input space and $Y = \{0, 1, ..., L-1\}$ is the set of L possible labels. We have two different distributions over $X \times Y$, called the source domain D_S and the target domain D_T. The unsupervised domain adaptation setting is then provided with a labeled source sample S drawn from D_S and an unlabeled target sample T drawn from D_T^X, where D_T^X is the marginal distribution of D_T over X.

$$S = (x_i, y_i)_{i=1}^{n} \sim (D_S)^n; \qquad T = (x_i)_{i=n+1}^{N} \sim (D_T^X)^{n'} \tag{4.1}$$

with $N = n + n'$ being the total number of samples.
The goal of the algorithm is to build a classifier $\eta : X \rightarrow Y$ with a low target error while having no information about the labels of D_T.

4.4 Categories

Research in unsupervised domain adaptation for NLP can be categorized in three main categories. Pivot-based approaches, loss-based approaches and data-based approaches. The most used methods for unsupervised domain adaptation are based on domain adversarial training. Another category of methods are those using pivots, cross-domain shared features, to construct a common feature space across domains. As data-based approaches are considered those using pseudo-labels to annotate data or selecting most relevant data points or exploiting the usage of pre-trained language models. For a similar categorization one could consider a recent survey [72].

4.5 Model-based approaches

4.5.1 Pivot-based approaches

Pioneering pivot-based methods for domain adaptation are structural correspondence learning (SCL) [15] and spectral feature alignment (SFA) [65]. The aim is to find features which are common across domains by using unlabeled data from both domains.

SCL uses auxiliary functions inspired by Ando and Zhang [6]. The first step of SCL is to define a set of pivot features to learn a mapping from the original feature spaces of both domains to a shared low-dimensional space. A high inner product in this new space indicates a high degree of correspondence. During training, both transformed and original features are used from the source domain. During testing, both transformed and original features are used from the target domain. Pivot features should occur frequently in both domains, in order to calculate the co-variance with non-pivot features accurately, but they must also be diverse to characterize the task. In practice there are many features that can serve as pivots, but only some are chosen. For each pivot feature a binary classification task is formed, to predict if that exact pivot is part of the instance, given that the feature itself is hidden. Since pivot features occur frequently in both domains, non-pivot features from both domains correlate with them. If two non-pivot features are correlated in the same way with many of the same pivot features, they have a high degree of correspondence.

A recent line of work by Ziser and Reichart [105], [107], [106], [108], brings SCL in a neural network context. Initially an auto-encoder structual correnspodence learning (AE-SCL) was proposed, that combines the pivot-based approach with auto-encoder neural networks. A three layer feed-forward network is used to learn hidden representations and map non-pivots to pivots, and those encodings are used to augment the training data. Moreover, given that some pivot features are similar to each other and are in conflict with others, it is proposed to use a pre-trained word embedding model as initialization for the auto-encoder that maps inputs to the hidden layer (AE-SCL-SR). These embeddings are induced by word2vec [60] trained with unlabeled data from the source and the target domains. The main drawback of the method, as also in previous pivot-based methods, is that a single, structure-indifferent and not context-dependent feature representation is learned.

In order to create more accurate representations, SCL is combined with neural language models and pivot-based language modeling (PBLM) is introduced by Ziser and Reichart [106] [107]. PBML consists of a randomly initialized embedding layer and an LSTM layer. PBML operates similarly to language models, the basic difference among them is the prediction they make over a given input. While a language model aims to predict the next input word, PBLM predicts the next word (or bigram) if it is a pivot, and NONE otherwise. Also it is introduced a k-order PBLM, that predicts the sequence of the next k words, as long as the sequence includes a pivot or NONE otherwise. After being trained the PBLM, without the top layer, is combined with LSTM or CNN layers. PBLM-LSTM is therefore a three-layer model, with the third layer -an LSTM layer- being feed from the PBLM's hidden representations. PBLM-CNN is a similar combination, but the PBLM's hidden representations are fed to a CNN layer. A two-step training procedure is employed. First, PBLM is trained with unlabeled data from both domains. Afterwards, the final model is trained with the source domain labeled data to preform the classification task. During that step the parameters of the pre-trained PBLM are held fixed.

A major weakness of pivot-based language model is the need of a huge number of pivot features. To tackle this issue, Ziser and Reichart [108] adopted a task refinement learning approach using PBLM, called TRL-PBLM. The TRL scheme gradually exposes pivots to PBLM. Pivots are divided in two subsets, those that in the source domain are more frequent in instances with positive labels and those that in the source domain are more frequent in instances with negative labels. PBLM is trained as in previous work on all unlabeled data. The main difference is that in cases where the next unigram or bigram is a pivot, instead of predicting the pivot identity, PBML should predict a positive or negative label, according the subset that the pivot is part of. After that initial step is completed, TRL algorithm continues for a predefined number of iterations. At each iteration the PBLM, initialized with the weights computed on the previous iteration, is exposed to an additional proportional part of pivots. In the last iteration all pivots are exposed. Pivots are given in a predefined sorted sequence, according three ranking methods that are tested. Ranking by mutual information, ranking by frequency and ranking by similar frequencies protocols are considered.

A common issue with all above mentioned methods is that they indicate two independent steps one for representation learning and one for task learning. To face the issue Miller [62] suggested to train the two tasks jointly. The method takes the same model and configuration as AE-SCL of [105], and defines a joint loss function as follows in equation Equation 4.2. In the following, the representation is $h(x) = ReLU(W_h x)$, $W_h \in R^{d \times n}$, task prediction is $f_{task}(x) = \sigma(W_t h(x))$, $W(t) \in R^{1 \times d}$, pivot prediction is $f_{pivot}(x) = \sigma(W_p h(x)), W_p \in R^{p \times d}$, D_l is the labeled source data, D_a is all data, θ defines the model parameters, λ controls the weight of pivot prediction loss and ρ is a weight for the regularization term R.

$$\begin{aligned}\mathcal{L}(D;\theta) = &\sum_{x,y \in D_l} BCE(f_{task}(x), y) \\ &+ \lambda \sum_{x \in D_a} BCE(f_{pivot}(x), pivots(x)) \\ &+ \rho R(\theta)\end{aligned} \tag{4.2}$$

During training labeled source data and unlabeled data from both domains are feed in the model alternately. Validation set consists only of labeled source data.

In the direction of learning pivots and non-pivots, a recent line of work by Li et al., [50], [52], [54], [51], proposes to learn them automatically via attention. On [50], Adversarial Memory Network is presented, that consists of two parameter-shared memory networks, with one for sentiment classification and the other for domain classification. The two networks are jointly trained in a domain adversarial way, as in DANN, that will presented in the next section. Each memory network contains multiple hops, each of which consists of an attention layer and a linear layer. As not all words contribute equally in the representation of each example, the attention mechanism extracts important words for the task and aggregate those word's representations to form the output. On a similar work, [52], Hierarchical Attention Transfer Network (HATN) is proposed, that consists of two hierarchical attention networks, with one (P-net) aiming to find pivots and the other (NP-net) aligning the non-pivots by using pivots as a bridge. Firstly P-net conducts individual attention learning to provide positive and negative pivots and then both networks conduct joint attention in a way that HATN can simultaneously capture pivots and non-pivots.

4.6 Loss-based approaches

4.6.1 Domain Adversarial Training

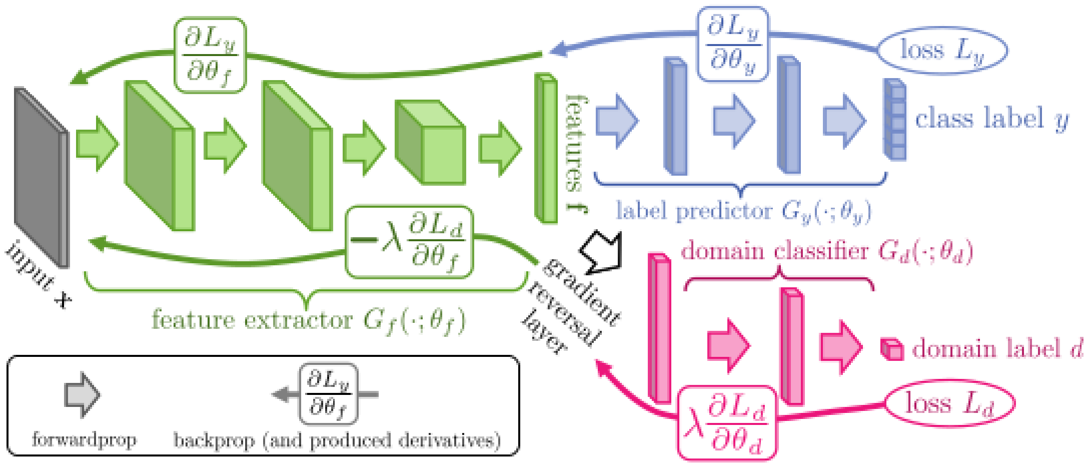
The most common methods for neural unsupervised domain adaptation are focused on the use of domain adversaries, as introduced by Ganin and Lempitsky [29], [30]. Inspired by how GANs [31] minimize the differences between training and synthetic distributions of data, domain adversarial training goal is to learn latent feature representations, that tend to reduce discrepancy of source and target distribution. The background behind this approach is based on the theory of domain adaptation introduced by Ben-Devid et al. [11], which states that cross-domain generalization can be achieved by extracting features that the domain origin of the input example cannot be identified.

The method boosts those features that are discriminative for the learning task on the source domain and indiscriminative with respect to the shift between the domains. Therefore, the goal is to calculate an accurate task predictor while maximizing the confusion of an auxiliary domain classifier in distinguishing source and target features. This is achieved by jointly optimizing the underlying features as well as two classifiers, operating on them, a label predictor for the classification task and a domain classifier that discriminates between the source and target domains during training. The task and domain classifiers are competing against each other, in an adversarial way. To do so, the domain classifier is connected to the feature extractor via a gradient reversal layer, that multiplies the gradient by a certain negative constant during the backpropagation-based training. Apart from that, training proceeds as usual and minimizes the label prediction loss for source examples and domain classification loss for all examples. Gradient reversal layer ensures that the feature distributions over the two domains are made similar, that is they are indistinguishable for the domain classifier. DANN architecture is presented in Figure 4.2.

Let $G_f(\cdot;\theta_f)$ be a feature extractor with parameters θ_f, that maps an example x (either source or target) in a hidden layer representation. Also, let $G_y(\cdot;\theta_y)$ be the part of the network that computes the label prediction output layer y, with parameters θ_y, while $G_d(;\theta_d)$ corresponds to the computation of the domain prediction output d of the network, with parameters θ_d. We can consider the composition of G_f and G_y as a classifier $F = G_f \circ G_y$ and the composition of G_f and G_d as a domain discriminator $D = G_f \circ G_d$. We will note the prediction loss and the domain loss respectively by:

$$L_y(\theta_h, \theta_y; D_S) = E_{x,y\sim D_S}[y^T lnF(x)] \tag{4.3}$$

$$L_d(\theta_h, \theta_d; D_S, D_T) = \sup_{\theta_d} E_{x\sim D_S}[lnD(x)] + E_{x\sim D_T}[ln(1 - D(x))] \tag{4.4}$$



$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
features f
class label y
label predictor $G_y(\cdot;\theta_y)$
$-\lambda\frac{\partial L_d}{\partial \theta_f}$
input x
gradient reversal layer
domain classifier $G_d(\cdot;\theta_d)$
feature extractor $G_f(\cdot;\theta_f)$
domain label d
loss L_d
$\lambda\frac{\partial L_d}{\partial \theta_d}$
forwardprop
backprop (and produced derivatives)

Figure 4.2: DANN architecture includes a deep feature extractor(green), a deep label predictor (blue), a domain classifier(red) and a gradient reversal layer. Figure from [30].

Therefore, domain adversarial training minimizes the objective

$$\min_{\theta_h,\theta_y} L_y(\theta_h,\theta_y;D_S) + \lambda_d L_d(\theta_h;D_S,D_T) \tag{4.5}$$

where λ_d is a weighting factor.

Domain Adversarial Neural Networks, is one of the most widely used domain adaptation approaches in NLP. They have been applied in a range of NLP applications in the last years mainly to sentiment classification [29], [52], and other tasks like language identification [47], relation extraction [28], [74], POS tagging [100], duplicate question detection [84], relevancy identification [4] and parsing [83].

Du et all. [27] brings domain adversarial training in the context of BERT models, proposes a novel domain-distinguish task, that predicts during pre-training if two sentences origin from the same target domain or from different domains. In addition during fine-tuning the proposed BERT based model is trained jointly with a domain adversarial loss via a GRL unit, as initialy proposed by Ganin et al. [30].

4.6.2 Reweighting

Reweighting is another approach for adaptation, that assigns an importance weight to each labeled example of the source domain in accordance with the similarity of the example to the target domain. Methods that explicitly reweight the loss based on domain discrepancy information include maximum mean discrepancy (MMD) by Gretton et al. [32] and its more efficient version called kernel mean matching (KMM)[33]. KMM reweights the training instances such that the means of the training and test points in a reproducing a kernel Hilbert space are close to each other. Jiang and Zhai [42] introduced instance weighting in NLP and proposed to learn weights by first training domain classifiers. In neural setups, an early study by Plank et al. [70] reports non-significant improvements for traditional part-of-speech (POS).

4.7 Data-based approaches

4.7.1 Pseudo-labeling

Pseudo-labeling, also known as bootstrapping, applies semi-supervised methods such as self-training [99], [58], co-training [16] and tri-training [104] by using the same model, multiple models or other more-accurate models as a guide to produce labels for the target data. Most of these works date back to traditional non-neural learning methods. This line of work was revised by Ruder and Plank [79], with neural models usage, showing that approaches like tri-training with neural models is a strong baseline for domain shift.

Self-training, Yarowsky [99], McClosky et al. [58], is one of the earliest and simplest bootstrapping methods. Fundamentally, it uses model's own predictions on unlabeled data to obtain additional information that can be used during training. Usually the most confident predictions are taken into account. Self-training trains a model m on a labeled training set L and an unlabeled data set U. At each iteration, the model provides predictions in the form of a probability distribution over classes for all unlabeled examples in U. If the probability assigned to the most likely class is higher than a predetermined threshold τ, the example is added to the labeled examples with the corresponding pseudo-label. Instead of using a fixed threshold, is is usual to to select the top n unlabeled examples that have been predicted with the highest confidence after every epoch and add them to the labeled data. The latest variant is called throttling as proposed by Abney [1].

Co-training, Blum and Mitchell [16], takes advantage of two separate views of data. It assumes that each example is described using two different feature sets that provide different, complementary information about the instance. The two views are considered as conditionally independent and sufficient for classification. Co-training first learns a separate classifier for each view using any labeled examples. The most confident predictions of each classifier on the unlabeled data are then used to iteratively construct additional labeled training data.

Tri-training, Zhi-Hua Zhou and Ming Li [104], utilizes the agreement of three independently trained models. Tri-training first trains three models m_1, m_2 and m_3 on samples of the labeled data. An unlabeled datum is added to the training set of a model in the other two models agree on its label. Training stops when classifiers do not change anymore.

Tri-training with disagreement, Søgaard [89], considers that a model will be strengthened in its weak points. To achieve this, it alternates the traditional tri-training algorithm, requiring that for a datum on which two models agree, the third model has to disagree on the prediction. Tri-training with disagreement is more data-efficient than tri-training.

On Saito et al. [80], classical tri-training has been modified, and asymmetric tri-training is proposed, in a way that two networks are used to learn the labels, and the third is trained in order to learn target-based representations. The proposed model includes a shared feature extractor F, two classifiers for labeled samples F_1, F_2 and a target-specific classifier F_t that learns from pseudo-labeled target samples. The method first trains the network from only labeled source samples and then pseudo-labels the target samples based on the output of the first two classifiers. The shared feature extractor learns from all classifiers F_1, F_2 and F_t. F_1 and $F2$ are expected to classify samples on different viewpoints, so a constraint is made for the weight of F_1, $F2$, to make their inputs different to each other. Assuming W_1 and W_2 are fully connected layers' weights of F_1, F_2, a term $||W_1^T W_2||$ is added to the cost function. In order to pick up reliable pseudo-labels the entire network is trained with source training set and then a label is given for each target element under the requirements that both F_1 and F_2 agree on the given pseudo-label and the maximizing probability of at least one of them exceeds a threshold parameter. A linearly growing candidate sampling scheme is proposed. After the pseudo-labeled training, F, F_1, F_2 are updated on the augmented training set and finally F and F_t are optimized.

On Ruder and Plank [79], a novel multi-task tri-training method is proposed. Based on multi-task training idea [21], replaces the three separate models, with models that share their parameters and are trained jointly as in multi-task learning. The output layers are model specific and are only updated for the inputs of the respective model. As the models learn a joint representation, is needed to obtain features as diverse as possible. In order to achieve the desired diversity, an orthogonality constraint is introduced as an additional loss term, as proposed by Bousmalis et al [17]. Given that $|| \cdot ||_F^2$ is the squared Forbenius norm and W_{m1} and W_{m2} are the output parameters of the two models, the constraint is defined as: $\mathcal{L}_{orth} = ||W_{m1}^{\tau} W_{m2}||_F^2$.

Data Selection is another data-based approach that tries to select the best matching data for the new domain [63], [7], [71]. This selection is done by using preplexity [63] or domain similarity measures such as Jensen-Shannon divergence [71], [73]. Data selection has been studied in machine translation [63], [94], [7], [3] and in parsing [71], [78].

4.7.2 Pretrained Language Models

Large pretrained transformer-based language models [39], [69], [25], are recently dominant in Natural Language Processing. Those models are commonly fine-tuned on down-stream tasks, with a small amount of labeled data and achieve high performance in a great variety of tasks. Pretraining and fine-tuning, can be seen as an adaptation procedure.

Taking the pretrained model and use it without any adaptation on the unseen data, can be considered as an extreme case, that is equivalent to zero-shot learning.

Using a domain specific pretrained model and fine-tune it on relevant data. Such approaches are developed for example in biomedical NLP, with BioBERT [46] trained on a corpus combination of general English documents and domain specific biomedical documents.

Using more unsupervised pretraining steps, which are gradually becoming more domain and task specific, describes a group of works, that can be called multi-phase pretraining. In Gururangan el al. [37] a second phase of pre-training in-domain (domain adaprive pre-training DAPT) is considered, as well as a task-adaptive pretraining (TAPT), in order to adapt to the task's unlabeled data.

An alternative is auxiliary task pre-training, this method is using relevant labeled auxiliary tasks in an multi-task learning set or in a supplementary training on labeled-data tasks for transfer.

Chapter 5

Unsupervised Domain Adaptation of BERT with domain pretraining and auxiliary masked language modeling

5.1 Introduction

In this work we explore Unsupervised Domain Adaptation (UDA) of pretrained language models for downstream tasks. We propose a fine-tuning procedure, using a mixed classification and Masked Language Model loss, that can adapt to the target domain distribution in a robust and sample efficient manner. We present evidence that the mixed loss can be effectively used for validation during UDA training. Furthermore, we conduct an analysis about the limitations of the dominant Domain Adversarial training for UDA based on theoretical concepts and experimental results. Our method is evaluated on twelve domain pairs of the Amazon Reviews Sentiment dataset, yielding 91.73% accuracy, which is a 1.10% absolute improvement over the current state-of-the-art.

Deep architectures have achieved state-of-the-art results in a variety of machine learning tasks. However, real world deployments of machine learning systems often operate under domain shift, which leads to domain degradation. This introduces the need for adaptation techniques, where a model is trained with data from a specific domain, and then can be optimized for use in new settings. Efficient techniques for model re-usability can lead to faster and cheaper development of machine learning applications and facilitate their wider adoption. Especially techniques for Unsupervised Domain Adaptation (UDA) can have high real world impact, because they do not rely on expensive and time-consuming annotation processes to collect labeled data for domain-specific supervised training, further streamlining the process.

UDA approaches in the literature can be grouped in three major categories, namely pseudo-labeling techniques (e.g. [103]), domain adversarial training (e.g. [30]) and pivot-based approaches (e.g. [15, 65]). Pseudo-labeling approaches use a model trained on the source labeled data to produce pseudo-labels for unlabeled target data and then train a model for the target domain in a supervised manner. Domain adversarial training aims to learn a domain-independent mapping for input samples by adding an adversarial cost during model training that minimizes the distance between the source and target domain distribution. Pivot-based approaches aim to select domain-invariant features (pivots) and use them as a basis for cross-domain mapping. This work does not fall under any of these categories, rather we aim to optimize the fine-tuning procedure of pretrained language models (LMs) for learning under domain-shift.

Transfer learning from language models pretrained in massive corpora[24, 39, 98, 56, 19] is a game changing progress in NLP which has yielded significant improvements across tasks, even when small amounts of data are used for fine-tuning. The fine-tuning procedure introduces the need for sequential transfer learning, that is adapting pretrained models to target tasks and new domains. Fine-tuning of pretrained LMs provides therefore a straightforward framework for semi-supervised domain adaptation that, although learning tasks from labeled data, exploits the dynamics of language models as unsupervised multitask learners. Still, besides the multi-domain knowledge encoded in a modern pretrained model, learning tasks from out-of-domain data remains a challenging issue.

In this work, we propose a fine-tuning method for BERT [24] in order to address the UDA problem. Our method is based on simultaneously learning the task from labeled data in the source distribution, while adapting to the language in the target distribution using multitask learning. The key idea of our method is that by simultaneously minimizing a task-specific loss on the source data and a language modeling loss on the target data during fine-tuning the model will be able to adapt to the language of the target domain, while learning the supervised task from the available labeled data.

Our key contributions are: (a) We propose a novel, simple and robust unsupervised domain adaptation procedure for downstream BERT models based on multitask learning, (b) we achieve state-of-the-art results for the Amazon reviews benchmark dataset, surpassing more complicated approaches and (c) we conduct a discussion on the limitations of adversarial domain adaptation, grounded on theoretical concepts and our empirical observations.

5.2 Related Work

Ben-David et al. [12, 10] provide a theory for learning from different domains. Ganin et al [30] and Ganin and Lempitsky [29] propose to learn a task while not being able to distinguish if samples come from the source or the target distribution, through use of an adversarial cost. According to Ramponi and Plank [72] domain adversarial training is a dominant approach for UDA in literature (Li et al., Alam et al., Sano et al.) [48, 5, 82]. Recently, Du et al. [26] pose domain adversarial training in the context of BERT models. Zhao et al. [102] propose multi-source domain adversarial networks. Guo et al. [35] propose a mixture-of-experts approach for multi-source UDA. Guo et al. [34] explore distance measures as additional losses and a dynamic bandit controller of domains. Shen et al. [86] learn domain invariant features via Wasserstein distance. Bousmalis et al. [18] introduce domain seperation networks with private and shared encoders.

Another line of UDA research includes pivot-based methods, focusing on extracting cross-domain features. Structural Correspondence Learning (SCL) (Blitzer et al. [15]) and Spectral Feature Alignment (Pan et al. [65]) were among the first pivot-based techniques. Ziser and Reichart [105, 106, 108] combine SCL with neural network architectures and language modeling. Miller [62] proposes to jointly learn the task and pivots. Li et al. [53] learn pivots with hierarchical attention networks. Pivot-based methods have also been used in conjunction with BERT (Ben-David et al. [9]).

Pseudo-labeling techniques are semi-supervised algorithms that either use the same model in self-training (Yarowsky, McClosky et al., Abney [99, 59, 2]) or multiple bootstrap models in tri-training (Zhou and Li, Søgaard [103, 90]) as guide to label the target unlabeled data. Saito et. al [81] propose an asymmetric tri-training approach. Ruder and Plank [79] introduced a multi-task tri-training method. Rotman and Reichart [76] and Lim et al. [55] study pseudo-labeling with contextualized word representations. Ye et al. [101] combine self-training pseudo-labeling with pretrained language models (XLM-R, Conneau et al. [23]) and propose CFd, class aware feature self-distillation.

Pretraining and fine-tuning is a plain and uncomplicated adaptation approach. When sufficient amount of data are available on a domain or application specific corpus, pretrain a model from scratch is a simple and effective idea. Domain specific models with BERT architecture like BioBERT (Lee et al. [45]) and SciBERT (Beltagy et al. [8]) do so. Sun et al. [91] propose further pretraining of BERT with target domain data and multi-tasking with relevant tasks for BERT fine-tuning. Xu et al. [97] introduce a review reading comprehension task and a post-training approach for BERT with an auxiliary loss on a question-answering task. Continue pretraining on multiple phases, from general to domain specific (DAPT) and task specific data (TAPT), further improve performance of PLMs, as shown by Gururangan et al. [36]. AdaptaBERT (Han and Eisenstein [38]) propose a second phase of unsupervised pretraining of a BERT based system in a unsupervised domain adaptation context.

Recent works have highlighted the merits of using Language Modeling as an auxiliary task during fine-tuning. Chronopoulou et al. [22] use an auxialiary LM loss to avoid catastrophic forgetting in transfer learning and Jia et al. [40] adopt this approach for cross-domain named-entity recognition.

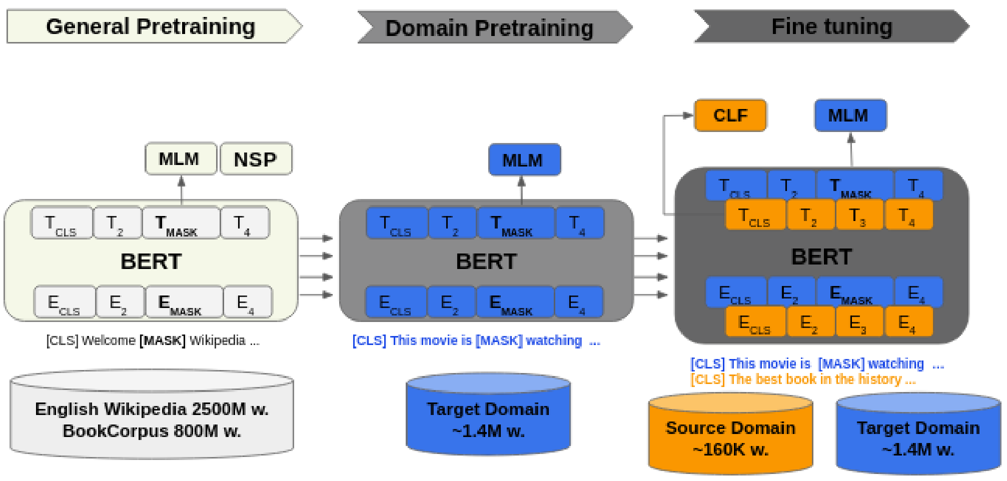

General Pretraining
Domain Pretraining
Fine tuning
MLM
NSP
BERT
[CLS] Welcome [MASK] Wikipedia ...
English Wikipedia 2500M w.
BookCorpus 800M w.
MLM
BERT
[CLS] This movie is [MASK] watching ...
Target Domain
~1.4M w.
CLF
MLM
BERT
[CLS] This movie is [MASK] watching ...
[CLS] The best book in the history ...
Source Domain
~160K w.
Target Domain
~1.4M w.

Figure 5.1: a) BERT [24] is pretrained on English Wikipedia and BookCorpus with the Masked Language Modeling (MLM) and the Next Sentence Prediction (NSP) tasks. (b) We continue pretraining BERT on unlabeled target domain data with the MLM task. (c) We fine-tune the model using both a classification loss on the labeled source data and MLM loss on the unlabeled target data.

5.3 Problem Definition

Let X be the input space and Y the set of labels, for binary classification tasks $Y = \{0, 1\}$. In domain adaptation there are two different distributions over $X \times Y$, called the source domain D_S and the target domain D_T. In the unsupervised setting labels are provided for samples drawn from D_S, while samples drawn from D_T are unlabeled. The goal is to train a model that performs well on samples drawn from the target distribution D_T. This is summarized in the following Equation:

$$\begin{aligned} S &= (x_i, y_i)_{i=1}^{n} \sim (D_S)^n \\ T &= (x_i)_{i=n+1}^{n+m} \sim (D_T^X)^m \end{aligned} \tag{5.1}$$

where D_T^X is the marginal distribution of D_T over X, n is the number of samples from the source domain and m is the number of samples from the target domain.

5.4 Proposed Method

Figure 5.1 gives an overview of our approach to Unsupervised Domain Adaptation, consisting of two steps. Starting from a pretrained model, we keep pretraining it on target domain data by the masked language modeling task. On a final fine-tuning step we learn the task on source labeled data, while we keep an auxiliary masked language modeling objective on unlabeled target data.

Figure 5.1 gives an overview of the proposed method. Starting from a pretrained in general corpora model, we keep pretraining it on target domain data by the masked language modeling task. On the final fine-tuning step we update the model weights using both a classification loss on the labeled source data and Masked Language Modeling loss on the unlabeled target data.

BERT [24] is based on the Transformer architecture [92]. It is trained as a Masked Language Model (MLM), by predicting masked words in the input. Additionally it uses a Next Sentence Prediction (NSP) loss, which classifies whether the pair of input sentences are continuous or not. During BERT pretraining, input tokens are randomly selected to be masked. The MLM task consists of predicting the most probable tokens for the masked positions. For fine-tuning, if a labeled dataset is available, a pretrained BERT model can be fine-tuned for the downstream task in a supervised manner with the addition of an output layer.

We initialize a model using the weights of a generally pretrained BERT and continue pretraining on an unsupervised set of in-domain data, in order to adapt to the target domain. This step does not require use of supervised data, we continue using the MLM objective for training.

For the final fine-tuning step, shown in we perform supervised fine-tuning on the source data, while we keep the MLM objective on the target data as an auxiliary task. Following standard practice, we use the `[CLS]` token representation for classification. The classifier consists of a single feed-forward layer.

During this procedure the model learns the task through the classification objective using the labeled source domain samples, and simultaneously it adapts to the target domain data through the MLM objective. The model is trained on the source domain labeled data for the classification task and target domain unlabeled data for the masked language modeling task. We mask only the target domain data. During training we interleave source and target data and feed them to the BERT encoder. Features extracted from the source data are then used for classification, while target features are used for Masked Language Modeling.

The joint loss is the sum of the classification loss L_{CLF} and the auxiliary MLM loss L_{MLM}. L_{CLF} is calculated on labeled examples from source domain, while L_{MLM} is calculated on unlabeled examples from target domain. We train the model over mixed batches, that include both source and target data, used for the respected tasks.

The mixed loss is presented in Eq. 5.2.

$$L(\mathbf{s}, \mathbf{t}) = \lambda L_{CLF}(\mathbf{s}) + (1 - \lambda) L_{MLM}(\mathbf{t}) \tag{5.2}$$

We give n labeled source samples $\mathbf{s} \sim D_S$ and m unlabeled target samples $\mathbf{t} \sim D_T$ on a batch. The weighting factor λ is selected as the proportion of labeled source data over the sum of labeled source and unlabeled target data, as stated in Eq. 5.3.

$$\lambda = \frac{n}{n + m} \tag{5.3}$$

5.5 Experiments

5.5.1 Dataset

We evaluate our method on the Amazon reviews multi-domain sentiment dataset [14], a standard benchmark dataset for domain adaptation. Reviews with one or two stars are labeled as negative, while reviews with four or five stars are labeled as positive. The dataset contains reviews on four domains: *Books* (B), *DVDs* (D), *Electronics* (E) and *Kitchen appliances* (K), yielding 12 adaptation scenarios of source-target domain pairs. Balanced sets of 2000 labeled reviews are available for each domain. We also use 19809 Books, 19798 DVDs, 19937 Electronics and 17805 Kitchen unlabeled reviews. For validation we use 20% of both labeled source and unlabeled target data. When a domain is considered as target for a particular adaptation scenario, all the available labeled data from this domain are used for testing.

5.5.2 Implementation Details

We use $BERT_{BASE}$ (uncased) as the Language Model on which we apply domain-specific pretraining. The $BERT_{BASE}$ original English model is a 12-layer, 768-hidden, 12-heads, 110M parameter neural network architecture, trained on the BooksCorpus with 800M words and a version of the English Wikipedia with 2,500M words. We follow the original proposed masking procedure, that is randomly masking 15% of WordPiece tokens [96]. If a token in a specific position is selected to be masked 80% of the time is replaced with a $[MASK]$ token, 10% of the time with a random token and 10% of the time remains unchanged. The maximum sequence length is set to 512 by truncation of inputs. During extended domain pretraining we train with batch size of 8 for 3 epochs. During the fine-tuning step we train with batch size of 36, consisting of 1 source sub-batch of 4 samples and 8 target sub-batches of 4 samples each. We update parameters after every 5 accumulated sub-batches.

For the domain adversarial experiment we set the weight factor of Eq. 5.4 to 0.01. We also experimented with $\lambda = 1$, $\lambda = 0.1$ and a sigmoid schedule for λ. We report best results. We do not use an early stopping criterion, as we found it harming for the performance, instead we train for fixed 10 epochs.

Models are developed with PyTorch [66] and HuggingFace Transformers [95].

5.5.3 Baselines - Compared methods

We select three state-of-the-art methods for comparison. Each of the selected methods represents a different line of UDA research, namely domain-adversarial training **BERT-DAAT** [26], self-training XLM-R based **p+CFd** [101] and pivot-based **R-PERL** [9]. Moreover we report results for the following settings:

- **Source only**: We fine-tune BERT on source domain labeled data, without using target data.
- **Domain Pretraining (DPT)**: We use the target domain unlabeled data in order to continue pretraining of BERT with MLM loss and then fine-tune the resulting model on source domain labeled data. data.
- **Domain Adversarial (DAT)**: Domain Adversarial Training with BERT. Starting from the domain pretrained BERT, we then fine-tune the model with domain adversarial training as in ganin2016domain. For a BERT model with parameters θ, with L_{CLF} being the prediction loss, L_{ADV} being the domain loss and λ_d being a weighting factor, domain adversarial training consists of the minimization criterion described in Eq. 5.4.

$$\min_{\theta} L_{CLF}(\theta; D_S) - \lambda_d L_{ADV}(\theta; D_S, D_T) \tag{5.4}$$

- **Proposed**: The proposed method, where we fine-tune the model created in the domain pretraining step using the mixed loss in Eq. 5.2.

5.6 Results and Discussion

	R-PERL	DAAT	p+CFd	BERT SO	BERT DAT	BERT DPT	Proposed
$B \rightarrow D$	87.8%	90.9%	87.7%	90.5%	90.7%	90.7%	**91.3%**
$B \rightarrow E$	87.2%	88.9%	**91.3%**	**91.3%**	91.1%	90.9%	91.2%
$B \rightarrow K$	90.2%	88.0%	92.5%	91.6%	92.8%	92.3%	**92.9%**
$D \rightarrow B$	85.6%	89.7%	**91.5%**	90.2%	90.6%	90.5%	91.4%
$D \rightarrow E$	89.3%	90.1%	91.6%	88.5%	88.8%	91.7%	**92.9%**
$D \rightarrow K$	90.4%	88.8%	92.5%	90.5%	92.0%	92.0%	**94.3%**
$E \rightarrow B$	90.2%	89.6%	88.7%	87.8%	89.4%	88.3%	**90.6%**
$E \rightarrow D$	84.8%	**89.3%**	88.2%	87.2%	86.5%	87.3%	88.4%
$E \rightarrow K$	91.2%	91.7%	93.6%	92.8%	94.6%	94.1%	**94.8%**
$K \rightarrow B$	83.0%	**90.8%**	89.8%	88.6%	83.6%	89.4%	89.4%
$K \rightarrow D$	85.6%	**90.5%**	87.8%	87.1%	83.6%	88.0%	89.2%
$K \rightarrow E$	91.2%	93.2%	92.6%	91.9%	92.4%	93.1%	**94.3%**
Average	87.50%	90.12%	90.63%	89.83%	89.68%	90.69%	**91.73%**

Table 5.1: Accuracy of domain adaptation on twelve domain pairs of Amazon Reviews Multi Domain Sentiment Dataset.

5.6.1 Comparison to state-of-the-art

We present results for all 12 domain adaptation settings in Table 5.1. The last line of Table 5.1 contains the macro-averaged accuracy over all domain pairs. The proposed method surpasses all other techniques, yielding an absolute improvement of 1.90% over the SO BERT baseline. For fair comparison, we compare only with methods based on pretrained models, mostly BERT. We observe that BERT fine-tuned only with the source domain labeled data, without any knowledge of the target domain, is a competitive baseline. This source-only model even surpasses state-of-the-art methods developed for UDA, e.g. R-PERL [9].

We try to reproduce the domain adversarial training procedure and present results in the DAT BERT column of Table 5.1. Adversarial training proved to be unstable in our experiments. After extensive hyper-parameter tuning of the adversarial loss weighting factor λ_d, we report the best results. We note that adversarial training does not manage to outperform the source-only baseline.Note that we did not have to perform extensive tuning for the other methods, including the proposed.

Domain pretraining increases the average accuracy with an absolute improvement of 0.86% over the source-only baseline. Continuing MLM pretraining on the target domain data leads to better model adaptation, and therefore improved performance, on the target domain. This is consistent with previous works on supervised [36, 97, 91] and unsupervised settings [38, 26].

The proposed method aims to learn the task (i.e. sentiment classification) from the available labeled data on the source domain, while at the same time adapting to the language of the target domain. This yields an additional 1.04% absolute improvement of average accuracy over the extended domain pretraining. Keeping the MLM loss during fine-tuning therefore, leads to better adaptation and acts as a regularizer that prevents the model from overfitting on the source domain.

The proposed method, surpasses in terms of macro-average accuracy all other proposed approaches for unsupervised domain adaptation on the Amazon reviews multi-domain sentiment dataset. Specifically, our method is by 1.08% better than the previous state-of-the-art gained by p+CFd [101], which is the best pseudo-labeling approach, by 1.59% DAAT [26] which is the best proposed domain adversarial method and by 4.21% R-PERL [9].

5.6.2 Sample efficiency

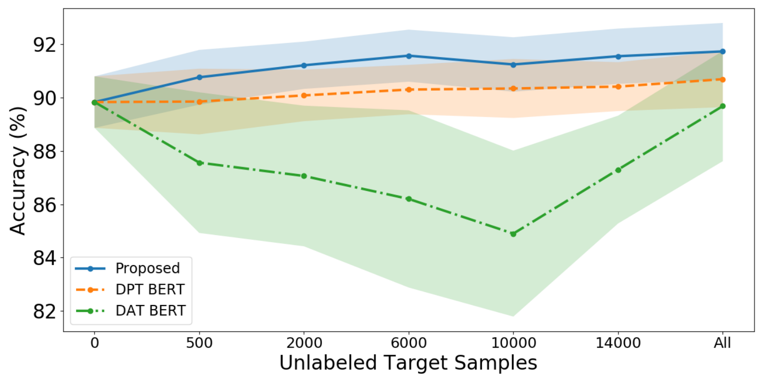


Figure 5.2: Average accuracy for different amount of target domain unlabeled samples of: (1) Domain Pretraining (2) Domain Adversarial Training and (3) Proposed.

We further investigate the impact of using different amount of target domain unlabeled data on model performance, to study the sample efficiency of the proposed method. We experiment with settings of 500, 2000, 6000, 10000 and 14000 samples, by randomly limiting the number of unlabeled target domain data. For each setting we conduct three experiments with BERT models: (1) DPT, (2) DAT and (3) the proposed. When no target data are available, all methods are equivalent to a source only fine-tuned BERT. Again, we do not tune the hyper-parameters for DPT or the proposed method. Fig. 5.2 shows the average accuracy on the twelve adaptation scenarios of the studied dataset. We see that the proposed method produces robust performance improvement when we limit the amount of target data, indicating that it can be used in low-resource settings. However, training BERT in a domain adversarial manner shows instabilities. This is further discussed in Section 5.7.

5.6.3 On the stopping criteria for UDA training

A common problem when performing UDA is the lack of target labeled data that can be used for hyperparameter validation. For example, Ruder and Plank [79] use a small set of labeled target data for validation, putting the problem in a semi-supervised setting. When training under a domain shift, optimizing performance on the source data may not yield optimal performance for the target data.

Stopping Criterion	Epochs	Av. Acc.
Fixed	1	90.98
Fixed	3	91.65
Fixed	10	**91.75**
Min source loss	10, patience 3	91.30
Min mixed loss	10, patience 3	**91.73**

Table 5.2: Comparison of average accuracy for various validation settings.

To illustrate this, we examine if the minimization of the mixed loss can be used as a stopping criterion for UDA training. We compare five stopping criteria: (1) fixed training for 1 epoch, (2) fixed training for 3 epochs, (3) fixed training for 10 epochs, (4) stop when the minimum classification loss is reached for the source data and (5) stop when the minimum mixed loss (Eq. 5.2) is reached. For (4) and (5) we train for 10 epochs with patience 3. We report average accuracy of the five stopping criteria over the twelve adaptation scenarios of Amazon Reviews dataset on Table 5.2. Training for a fixed number of 10 epochs and stopping when the minimum mixed loss perform best, yielding comparable accuracies of 91.75 and 91.73 respectively. Note that stopping when the minimum source loss stops the fine-tuning process too soon and does not allow the model to learn the target domain effectively. Overall, we observe that the mixed loss can be effectively used for early stopping, regularizing the model and alleviating the need for extensive search for the optimal number of training steps. We postulate, this is an indication that the mixed loss can be successfully used for model validation and low mixed loss is indicative of successful model validation.

5.6.4 Visualization of features

We present t-SNE plots of the representations learned by different methods on $D \rightarrow K$ task in Figure 5.3. These are the $[CLS]$ token representations of labeled source domain training data and of target domain test data. In Figure 5.3a we see that domain adversarial training introduces significant distortion in the semantic space, which is reflected in model performance. In Figure 5.3c we present the representations created by directly fine-tuning $BERT_{BASE}$ on source domain data. As expected source positive and negative samples are well separated. On the contrary, target positive and target negative samples, that follow in general the respective source clusters, in addition share a common indistinguishable area. In Figure 5.3d we show the representations obtained from BERT that was firstly pretrained on target unlabeled data and then fine-tuned on source domain data. That second representation follows the previous pattern, although we observe that the indistinguishable area of target positive and negative data points is significantly reduced. In Figure 5.3a are the feature representations obtained by our proposed method. Blue and yellow data points are better separated than in the previous cases and the region where mix together is smaller. So, positive and negative samples from the target domain are well distinguished and part of corresponding clusters with training source data points. That is, the proposed model adapted in a better way to the unsupervised target domain task.

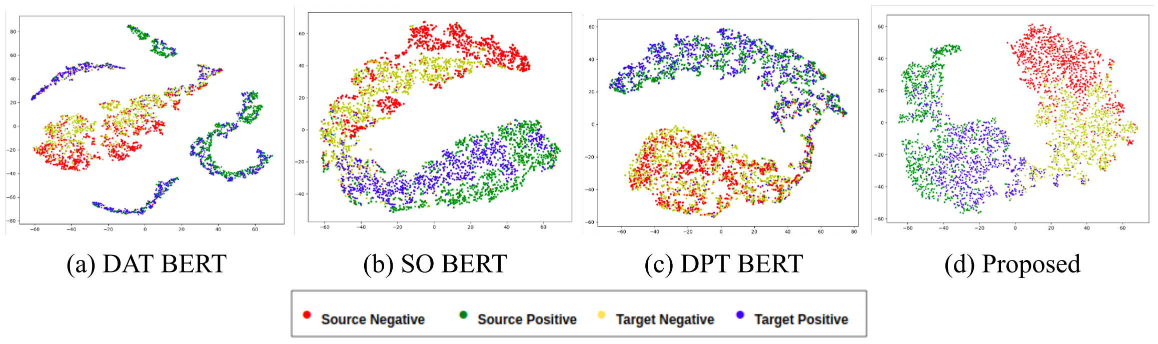


Figure 5.3: Visualization of BERT features for UDA: Reduced $2D$ representations of BERT $[CLS]$ features using t-SNE for the $D \rightarrow K$ task.

5.7 On the limitations of Domain Adversarial Training

5.7.1 Background Theory

Ben-David et al. [12, 10] present a theory of learning from different domains. A key outcome of their work is the following theorem.

Theorem 1 [12, 10] Let H be the hypothesis space and let D_S, D_T be the two domains and ϵ_S, ϵ_T be the corresponding error functions. Then for any $h \in H$:

$$\epsilon_T(h) \leq \epsilon_S(h) + \frac{1}{2} d_{H\Delta H}(D_S, D_T) + C \tag{5.5}$$

where $d_{H\Delta H}(D_S, D_T)$ is the $H\Delta H$-divergence [43] between the two domains, that is a measure of distance between domains that can be estimated from finite samples.

Eq. 5.5 defines an upper bound for the expected error $\epsilon_T(h)$ of a hypothesis h on the target domain as the sum of three terms, namely the expected error on the source domain $\epsilon_S(h)$, the divergence between the source and target domain distributions $\frac{1}{2} d_{H\Delta H}(D_S, D_T)$ and the error of the ideal joint hypothesis C. When such an hypothesis exists, the term is considered relatively small and in practice ignored. The first term, bounds the expected error on the target domain by the expected error in the source domain and is expected to be small, due to supervised learning on the source domain. The second term, gives a notion of distance between the source and target domain extracted features. Intuitively this Equation states: “if there exists a hypothesis h that has small error on the source data and the source feature space is close to the target feature space, then this hypothesis will have low error on the target data”. Domain Adversarial Training aims to learn features that simultaneously result to low source error and low distance between target and source feature spaces based on the combined loss in Eq. 5.4.

5.7.2 A-distance only provides an upper bound for target error

According to [12] the $H\Delta H$-divergence can be approximated by proxy A-distance, that is defined by Equation 5.6 given the domain classification error e.

$$d_A = 2(1 - 2\min\{e, 1 - e\}) \tag{5.6}$$

We calculate an approximation of the distance between domains. Following prior work [30, 80] we create an SVM domain classifier. We feed the SVM with BERT’s $[CLS]$ token representations, measure the domain classification error, and compute A-distance as in Eq. 5.6. We train the domain classifier on 2000 samples from each source and target domains. Fig. 5.4 shows the A-distance in comparison to the target error, averaged over the twelve available domain pairs using representations obtained from four methods, namely BERT SO, DAT BERT, DPT BERT and the proposed method. DAT BERT minimizes the distance between domains. DPT BERT also reduces the A-distance, to similar levels with the domain adversarial model, without aiming to achieve so. To our surprise we found that, although it achieves the lowest error rate, the proposed method does not significantly reduce the proxy A-distance compared to the source-only baseline.

Therefore, lower distance between domains, achieved intentionally or not, does not lead to better performance on the target domain. Additionally, optimizing performance on the target domain does not require minimizing the distance between domains. Overall we do not observe any correlation between the resulting A-distance and model performance on target domain. The above confirms the results of [88], that states if the feature extraction function has high-capacity then domain adversarial training is not sufficient for domain adaptation.

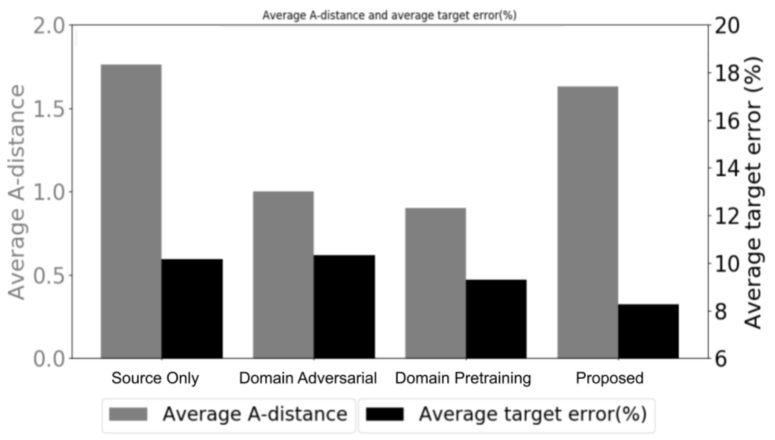


Figure 5.4: Comparison of average A-distance and average target error rate of different methods over all source - target pairs of the Amazon reviews dataset.

Domain adversarial trained BERT minimizes the distance between domains. Extended domain pretrained BERT also reduces the A-distance, to similar levels with the domain adversarial model, without aiming to achieve so. We surprisingly found that, although extended domain pretraining reduces distance between domains, our proposed method does not significantly reduce the proxy A-distance compared to the source-only baseline. We could assume that the $BERT_{BASE}$ model is of enough capacity to solve both the desired task on target domain and the domain classification task. Therefore, lower distance between domains, achieved intentionally or not, does not lead to better performance on the target domain. Additionally, optimizing performance on the target domain does not require minimizing the distance between domains. Overall we do not observe any correlation between the resulting A-distance and model performance on target domain.

Therefore, lower distance between domains, achieved intentionally or not, does not lead to better performance on the target domain. Additionally, optimizing performance on the target domain does not require minimizing the distance between domains. Overall we do not observe any correlation between the resulting A-distance and model performance on target domain.

5.7.3 Instability of Domain Adversarial Training

Domain adversarial training [30] faces some critical limitations that make the method difficult to be reproduced due to high hyper-parameter sensitivity and instability during training.

Such limitations have been highlighted by other authors in the UDA literature. For example, according to Shen et al. [86] when a domain classifier can perfectly distinguish target from source representations, there will be a gradient vanishing problem. Shah et al. [85] state that domain adversarial training is unstable and needs careful hyper-parameter tuning for their experiments. Wang et al. [93] report results over three multi-domain NLP datasets, where domain adversarial training in conjunction with BERT under-performs. Ruder and Plank [79]found that the domain adversarial loss did not help for their experiments on the Amazon reviews dataset.

In our experiments we note that domain-adversarial training results to worse performance than source only training. Furthermore, we experienced the need for extensive tuning of the λ_d parameter from Eq. 5.4 every time the experimental setting changed (e.g. when testing for different amounts of available target data). This can be seen in Figure 5.3, where we see visualizations of BERT features for UDA performed using Domain adversarial training Figure 5.3a, source only Figure 5.3b, domain pretraining Figure 5.3c and the proposed method Figure 5.3d. We see that domain adversarial training introduces significant distortion in the semantic space, which is reflected in model performance.

One could easily observe that the maximization of the L_{ADV} term could be achieved by predicting the wrong domain label for each example given to the domain discriminator. Specifically, there is a failure condition where the model just flips all the domain labels in order to maximize L_{ADV}. Such a prediction still makes the learned features domain-dependent, as the model achieves distinguishing domains, and only fails to name the proper label. Empirically we observed that kind of behavior when trying to train BERT on a domain adversarial manner, and only extensive hyper-parameter tuning could alleviate this issue.

Therefore, given that the there are no strict theoretical guarantees and the practical considerations, domain adversarial training may not be the best approach in every UDA scenario.

5.8 Conclusions and Future Work

In this work we explore unsupervised domain adaptation for sentiment classification and propose a new method to address the issue. We found that the well-spread domain adversarial technique facing theoretical and empirical limitations in the era of high-capacity transformer based pretrained language models, leading to training instabilities and to no improvement for the studied task. We propose a method, consisting of two steps, extended domain-specific pretraining and fine-tuning with auxiliary masked language modeling. Experiments on the well studied benchmark dataset of multi-domain Amazon reviews, yield remarkably improved results, that outperform state-of-the art-methods from all long-established approaches to the matter.

The proposed method could also be applied to other tasks under domain shift, such as sequence classification, question answering and part-of-speech tagging. Another field of application could be temporal and style adaptation. Moreover it would be of significant added-value to apply the method on applications over the observed deployment shift between labeled training and unlabeled real data. It would be of great importance to further investigate relevant auxiliary tasks for both pretraining and fine-tuning, as well as ways of combining all losses. In the future we would like to explore the applicability of our method to supervised scenarios, where the MLM auxiliary task could be used over unlabeled data coming from the same or a similar domains.

Chapter 6

Conclusions & Future Work

6.1 Conclusions

Unsupervised Domain Adaptation of pretrained language models is a challenging problem with direct real world applications. Besides the multi-domain knowledge encoded in a modern pretrained model, learning tasks from out-of-domain data remains a challenging issue.

Unsupervised Domain Adaptation approaches in the literature can be grouped in three major categories, namely pseudo-labeling techniques, domain adversarial training, and pivot-based approaches. This work is not part any of these categories, rather we aim to optimize the fine-tuning procedure of pretrained language models for learning under domain-shift. Pretraining and fine-tuning provide a straightforward framework for adaptation, while the usage of language modeling and its variants exploit the dynamics of models as unsupervised multitask learners.

In this work we propose a plug and play training strategy, which is able to improve performance in the target domain. That is fine-tuning BERT with a mixed classification and Masked Language Model loss, that can adapt to the target domain distribution in a robust and sample efficient manner. The proposed method aims to learn the task (i.e. sentiment classification) from the available labeled data on the source domain, while at the same time adapting to the language of the target domain. Keeping the MLM loss during fine-tuning, leads to better adaptation and acts as a regularizer that prevents the model from overfitting on the source domain

The proposed method achieves state-of-the-art results across 12 adaptation settings in the multi-domain Amazon reviews dataset. We observe that BERT fine-tuned only with the source domain labeled data, without any knowledge of the target domain, is a competitive baseline. We found continue pretraining in domain specific data to help adaptation in accordance to related previous work and we have incorporated this step into our approach. In contrast we found domain adversarial training to under-perform the baseline even after extensive parameter tuning. The proposed method, surpasses in terms of macro-average accuracy all other approaches for unsupervised domain adaptation representing respected lines of work and achieved state-of-the-art results, yielding a 1.1% absolute improvement of average accuracy.

When training under a domain sift, optimizing performance on the source distribution may not imply optimizing performance on the unlabeled target distribution. Our method produces robust results with little hyper-parameter tuning and the proposed mixed-loss can be used for model validation, allowing for fast model development and alleviating the need for extensive experimentation to find the optimal number of training steps.

Furthermore, the proposed method scales with the amount of available unsupervised data from the target domain, allowing for adaptation in low-resource settings. Even with little amount of target data keeping an MLM objective during fine-tuning leads to significantly improved adaptation.

We conclude this work by performing an discussion on the dominant Domain Adversarial training approach. We find that Domain Adversarial Training under-performs for our experiments, while it exhibits instability during training. We found minimizing A-distance between domains to be not sufficient when using a high capacity model like BERT. We do not observe any correlation between the resulting A-distance, that domain adversarial training minimizes, and model performance on target domain. Overall we faced various instabilities when training BERT in a domain adversarial manner, that are in line with observations mentioned in previous work. We propose that this instability results from the introduction of the adversarial term and can be alleviated only with careful hyper-parameter selection, while the under-performance can be due to the lack of strict theoretical guarantees of this method.

6.2 Future Work

In the future various extensions and variations of our work could be considered. We divide these works into three major categories. The first is about applicability of our work on domain adaptation on other tasks, beside sentiment analysis, and other types of adaptation. A second line of work could explore different settings for domain adaptation. A third category is related with the specific parts of our proposed method.

The proposed method could be applied to other tasks under domain shift. Evaluation on the benchmark Amazon reviews dataset yields promising results, nevertheless it would be useful to explore applicability to other tasks. Previous work evaluates domain adaptation primarily on sentiment analysis. Tasks also considered could be language identification [47], natural language inference [36, 75], part-of-speech tagging [38, 79, 67, 100], dependency parsing [82, 76, 49], name entity recognition [44, 41, 67] and relation extraction [28, 87].

To the above list of tasks we would add temporal and style adaptation. Language evolves over time, but used corpora consist exclusively of text written since the late 20th century. It is therefore crucial to determine whether pretrained models are transferable to texts from other periods or other stylistic traditions, such as historical documents. Temporal adaptation, to older or newer versions of specific data for which labels are unavailable. Even more, the method could be customized for style adaptation scenarios. Usually large labeled corpora are available on specific styles, for example Amazon reviews, although there is a need for high accuracy on other styles of text, as for example tweets or official documents.

Working on the development shift of data distributions would be an approach of high added-value. While the vast majority of deep learning systems are trained and evaluated on a specific data distribution, real world models are often used in out-of-domain settings which results in performance degradation. The used dataset simulates this situation but differs significantly. If available data were available testing the applicability of the proposed method to real scenarios faced by the industry would confirm the importance of the unsupervised domain adaptation research.

This work deals specifically with the unsupervised setting of domain adaptation, with a single source and a singe target, where labeled data are given for the source domain and unlabeled data for the target domain. In future we plan to expand our research to other settings. A multi-domain setting, where data from many sources are available, could be considered. Multi-source domain adaptation is a extension in which the labeled data may be collected from multiple sources with different distributions.

A common problem when performing UDA is the lack of target labeled data that can be used for hyperparameter validation. Semi-supervised settings, with few labeled data on target domain [79], specially used for validation purposes, could be considered. With such labeled samples available it is of added value to compare the performance of the proposed mixed loss as a stopping criterion with a classification loss on the target domain, calculated on the available validation set.

Moreover we are interested in source free settings for domain adaptation. In those settings the goal is to develop an accurate system for a target domain when annotations exist for a related domain but cannot be distributed and instead a model already trained on that annotations is available. Data sharing restrictions are common in datasets, specially in clinical NLP, where patient health information must be protected. Source free domain adaptation aims to develop intelligent systems in the face of data sharing constraints.

In addition, we plan to extended our work in adapting pretrained models in a supervised setting. In this work we found beneficial to keep an MLM objective during fine-tuning in order to achieve unsupervised domain adaptation. Usually some in-domain labeled data are available, while domain specific unlabeled data are easily accessible. Those unlabeled data could be used for an unsupervised task during fine-tuning, in a multitasking manner. Specifically we are interested in checking if auxiliary masked language modeling could serve as an effective regularizer during fine tuning of BERT or other pretrained models on in-domain settings. That could be the case in domains, where some labeled data are available but also are even more unlabeled data.

Finally, an interesting extension we would like to work on is auxiliary tasks and losses for both pretraining and fine tuning of language models. BERT is pretrained with the Masked Language Modeling and the Next Sentence Prediction Task. Some models such as RoBERTa [56] rely solely on a MLM variant, while others [93] incorporate one or more different auxiliary loss functions. Multitasking is mainly applied during pretraining, given the results of the present work using more tasks during fine-tuning is likely to yield positive results. Combining tasks is also a challenging issue during multitasking. Summing is a simple and uncomplicated idea but cannot accommodate tasks that require different input structures. Alternating losses or incrementally add tasks and summing the losses from all added tasks, could be investigated among other approaches. Overall we propose exploring possible auxiliary tasks and different ways to include multiple tasks into pretraining and fine-tuning.

Bibliography

[1] Steven Abney. *Semisupervised Learning for Computational Linguistics*. en. Google-Books-ID: VCd67cGB_rAC. CRC Press, Sept. 2007. isbn: 978-1-4200-1080-0.

[2] Steven Abney. *Semisupervised learning for computational linguistics*. CRC Press, 2007.

[3] Roee Aharoni and Yoav Goldberg. "Unsupervised Domain Clusters in Pretrained Language Models". In: *arXiv:2004.02105 [cs]* (May 2020). arXiv: 2004.02105. url: `http://arxiv.org/abs/2004.02105` (visited on 09/08/2020).

[4] Firoj Alam, Shafiq Joty, and Muhammad Imran. "Domain Adaptation with Adversarial Training and Graph Embeddings". In: *arXiv:1805.05151 [cs, stat]* (May 2018). arXiv: 1805.05151. url: `http://arxiv.org/abs/1805.05151` (visited on 08/29/2020).

[5] Firoj Alam, Shafiq Joty, and Muhammad Imran. "Domain adaptation with adversarial training and graph embeddings". In: *arXiv preprint arXiv:1805.05151* (2018).

[6] Rie Kubota Ando and Tong Zhang. "A framework for learning predictive structures from multiple tasks and unlabeled data". In: *Journal of Machine Learning Research* 6.Nov (2005), pp. 1817–1853.

[7] Amittai Axelrod, Xiaodong He, and Jianfeng Gao. "Domain Adaptation via Pseudo In-Domain Data Selection". In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, July 2011, pp. 355–362. url: `https://www.aclweb.org/anthology/D11-1033` (visited on 09/08/2020).

[8] Iz Beltagy, Kyle Lo, and Arman Cohan. "SciBERT: A Pretrained Language Model for Scientific Text". In: *arXiv:1903.10676 [cs]* (Sept. 2019). arXiv: 1903.10676. url: `http://arxiv.org/abs/1903.10676` (visited on 09/30/2020).

[9] Eyal Ben-David, Carmel Rabinovitz, and Roi Reichart. "PERL: Pivot-based Domain Adaptation for Pre-trained Deep Contextualized Embedding Models". In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 504–5221.

[10] Shai Ben-David et al. "A theory of learning from different domains". In: *Machine learning* 79.1-2 (2010), pp. 151–175.

[11] Shai Ben-David et al. "A theory of learning from different domains". en. In: *Machine Learning* 79.1-2 (May 2010), pp. 151–175. issn: 0885-6125, 1573-0565. doi: `10.1007/s10994-009-5152-4`. url: `http://link.springer.com/10.1007/s10994-009-5152-4` (visited on 08/29/2020).

[12] Shai Ben-David et al. "Analysis of representations for domain adaptation". In: *Advances in neural information processing systems*. 2007, pp. 137–144.

[13] Yoshua Bengio et al. "A neural probabilistic language model". In: *Journal of machine learning research* 3.Feb (2003), pp. 1137–1155.

[14] John Blitzer, Mark Dredze, and Fernando Pereira. "Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification". en. In: (), p. 8.

[15] John Blitzer, Ryan McDonald, and Fernando Pereira. “Domain adaptation with structural correspondence learning”. In: *Proceedings of the 2006 conference on empirical methods in natural language processing*. 2006, pp. 120–128.

[16] Avrim Blum and Tom Mitchell. “Combining labeled and unlabeled data with co-training”. en. In: *Proceedings of the eleventh annual conference on Computational learning theory - COLT’ 98*. Madison, Wisconsin, United States: ACM Press, 1998, pp. 92–100. isbn: 978-1-58113-057-7. doi: `10.1145/279943.279962`. url: `http://portal.acm.org/citation.cfm?doid=279943.279962` (visited on 09/04/2020).

[17] Konstantinos Bousmalis et al. “Domain Separation Networks”. In: *arXiv:1608.06019 [cs]* (Aug. 2016). arXiv: 1608.06019. url: `http://arxiv.org/abs/1608.06019` (visited on 09/08/2020).

[18] Konstantinos Bousmalis et al. “Domain separation networks”. In: *Advances in neural information processing systems*. 2016, pp. 343–351.

[19] Tom B Brown et al. “Language models are few-shot learners”. In: *arXiv preprint arXiv:2005.14165* (2020).

[20] Rich Caruana. “Multitask learning”. In: *Machine learning* 28.1 (1997), pp. 41–75.

[21] Rich Caruana. *Multitask Learning | SpringerLink*. 1997. url: `https://link.springer.com/article/10.1023/A:1007379606734` (visited on 09/08/2020).

[22] Alexandra Chronopoulou, Christos Baziotis, and Alexandros Potamianos. “An Embarrassingly Simple Approach for Transfer Learning from Pretrained Language Models”. In: *arXiv:1902.10547 [cs]* (Feb. 2019). arXiv: 1902.10547. url: `http://arxiv.org/abs/1902.10547` (visited on 03/15/2019).

[23] Alexis Conneau et al. “Unsupervised cross-lingual representation learning at scale”. In: *arXiv preprint arXiv:1911.02116* (2019).

[24] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *arXiv:1810.04805 [cs]* (Oct. 2018). arXiv: 1810.04805. url: `http://arxiv.org/abs/1810.04805` (visited on 03/15/2019).

[25] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[26] Chunning Du et al. “Adversarial and Domain-Aware BERT for Cross-Domain Sentiment Analysis”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 4019–4028. url: `https://www.aclweb.org/anthology/2020.acl-main.370` (visited on 07/16/2020).

[27] Chunning Du et al. “Adversarial and Domain-Aware BERT for Cross-Domain Sentiment Analysis”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 4019–4028.

[28] Lisheng Fu et al. “Domain Adaptation for Relation Extraction with Domain Adversarial Neural Network”. In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Taipei, Taiwan: Asian Federation of Natural Language Processing, Nov. 2017, pp. 425–429. url: `https://www.aclweb.org/anthology/I17-2072` (visited on 08/29/2020).

[29] Yaroslav Ganin and Victor Lempitsky. “Unsupervised Domain Adaptation by Backpropagation”. en. In: (), p. 10.

[30] Yaroslav Ganin et al. “Domain-Adversarial Training of Neural Networks”. In: *arXiv:1505.07818 [cs, stat]* (May 2015). arXiv: 1505.07818. url: `http://arxiv.org/abs/1505.07818` (visited on 08/29/2020).

[31] Ian J. Goodfellow et al. “Generative Adversarial Networks”. In: *arXiv:1406.2661 [cs, stat]* (June 2014). arXiv: 1406.2661. url: `http://arxiv.org/abs/1406.2661` (visited on 08/29/2020).

[32] Arthur Gretton et al. “A kernel method for the two-sample-problem”. In: *Advances in neural information processing systems*. 2007, pp. 513–520.

[33] Arthur Gretton et al. “Covariate shift by kernel mean matching”. In: *Dataset shift in machine learning* 3.4 (2009), p. 5.

[34] Han Guo, Ramakanth Pasunuru, and Mohit Bansal. “Multi-Source Domain Adaptation for Text Classification via DistanceNet-Bandits.” In: *AAAI*. 2020, pp. 7830–7838.

[35] Jiang Guo, Darsh J Shah, and Regina Barzilay. “Multi-source domain adaptation with mixture of experts”. In: *arXiv preprint arXiv:1809.02256* (2018).

[36] Suchin Gururangan et al. “Don’t Stop Pretraining: Adapt Language Models to Domains and Tasks”. In: *arXiv:2004.10964 [cs]* (May 2020). arXiv: 2004.10964. url: `http://arxiv.org/abs/2004.10964` (visited on 08/30/2020).

[37] Suchin Gururangan et al. “Don’t Stop Pretraining: Adapt Language Models to Domains and Tasks”. In: *arXiv preprint arXiv:2004.10964* (2020).

[38] Xiaochuang Han and Jacob Eisenstein. “Unsupervised Domain Adaptation of Contextualized Embeddings for Sequence Labeling”. In: *arXiv:1904.02817 [cs]* (Sept. 2019). arXiv: 1904.02817. url: `http://arxiv.org/abs/1904.02817` (visited on 09/29/2020).

[39] Jeremy Howard and Sebastian Ruder. “Universal Language Model Fine-tuning for Text Classification”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 328–339.

[40] Chen Jia, Xiaobo Liang, and Yue Zhang. “Cross-Domain NER using Cross-Domain Language Modeling”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 2464–2474. doi: `10.18653/v1/P19-1236`. url: `https://www.aclweb.org/anthology/P19-1236` (visited on 09/30/2020).

[41] Chen Jia, Xiaobo Liang, and Yue Zhang. “Cross-domain NER using cross-domain language modeling”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 2464–2474.

[42] Jing Jiang and ChengXiang Zhai. “Instance weighting for domain adaptation in NLP”. In: *Proceedings of the 45th annual meeting of the association of computational linguistics*. 2007, pp. 264–271.

[43] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. “Detecting change in data streams”. In: *VLDB*. Vol. 4. Toronto, Canada. 2004, pp. 180–191.

[44] Young-Bum Kim, Karl Stratos, and Dongchan Kim. “Adversarial adaptation of synthetic or stale data”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2017, pp. 1297–1307.

[45] Jinhyuk Lee et al. “BioBERT: a pre-trained biomedical language representation model for biomedical text mining”. In: *arXiv:1901.08746 [cs]* (Oct. 2019). arXiv: 1901.08746. doi: `10.1093/bioinformatics/btz682`. url: `http://arxiv.org/abs/1901.08746` (visited on 09/30/2020).

[46] Jinhyuk Lee et al. “BioBERT: a pre-trained biomedical language representation model for biomedical text mining”. In: *Bioinformatics* 36.4 (2020), pp. 1234–1240.

[47] Yitong Li, Timothy Baldwin, and Trevor Cohn. “What’s in a Domain? Learning Domain-Robust Text Representations using Adversarial Training”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 474–479. doi: `10.18653/v1/N18-2076`. url: `https://www.aclweb.org/anthology/N18-2076` (visited on 08/29/2020).

[48] Yitong Li, Timothy Baldwin, and Trevor Cohn. “What’s in a domain? learning domain-robust text representations using adversarial training”. In: *arXiv preprint arXiv:1805.06088* (2018).

[49] Zhenghua Li et al. “Semi-supervised domain adaptation for dependency parsing”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 2386–2395.

[50] Zheng Li et al. “End-to-End Adversarial Memory Network for Cross-domain Sentiment Classification”. en. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. Melbourne, Australia: International Joint Conferences on Artificial Intelligence Organization, Aug. 2017, pp. 2237–2243. isbn: 978-0-9992411-0-3. doi: `10.24963/ijcai.2017/311`. url: `https://www.ijcai.org/proceedings/2017/311` (visited on 09/04/2020).

[51] Zheng Li et al. “Exploiting Coarse-to-Fine Task Transfer for Aspect-Level Sentiment Classification”. en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019). Number: 01, pp. 4253–4260. issn: 2374-3468. doi: `10.1609/aaai.v33i01.33014253`. url: `https://www.aaai.org/ojs/index.php/AAAI/article/view/4332` (visited on 09/04/2020).

[52] Zheng Li et al. “Hierarchical Attention Transfer Network for Cross-Domain Sentiment Classification”. en. In: (), p. 9.

[53] Zheng Li et al. “Hierarchical attention transfer network for cross-domain sentiment classification”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[54] Zheng Li et al. “Transferable End-to-End Aspect-based Sentiment Analysis with Selective Adversarial Learning”. In: *arXiv:1910.14192 [cs]* (Oct. 2019). arXiv: 1910.14192. url: `http://arxiv.org/abs/1910.14192` (visited on 09/04/2020).

[55] KyungTae Lim et al. “Semi-Supervised Learning on Meta Structure: Multi-Task Tagging and Parsing in Low-Resource Scenarios.” In: *AAAI*. 2020, pp. 8344–8351.

[56] Yinhan Liu et al. “Roberta: A robustly optimized bert pretraining approach”. In: *arXiv preprint arXiv:1907.11692* (2019).

[57] Christopher Manning et al. *Language Models, RNN, GRU and LSTM*. 2019. url: `http://web.stanford.edu/class/cs224n/readings/cs224n-2019-notes05-LM_RNN.pdf`.

[58] David McClosky, Eugene Charniak, and Mark Johnson. “Effective Self-Training for Parsing”. In: *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. New York City, USA: Association for Computational Linguistics, June 2006, pp. 152–159. url: `https://www.aclweb.org/anthology/N06-1020` (visited on 09/04/2020).

[59] David McClosky, Eugene Charniak, and Mark Johnson. “Effective self-training for parsing”. In: *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. 2006, pp. 152–159.

[60] Tomas Mikolov et al. “Distributed representations of words and phrases and their compositionality”. In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[61] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[62] Timothy Miller. “Simplified Neural Unsupervised Domain Adaptation”. 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. 414–419. doi: `10.18653/v1/N19-1039`. url: `https://www.aclweb.org/anthology/N19-1039` (visited on 01/24/2020).

[63] Robert C. Moore and William Lewis. “Intelligent Selection of Language Model Training Data”. In: *Proceedings of the ACL 2010 Conference Short Papers*. Uppsala, Sweden: Association for Computational Linguistics, July 2010, pp. 220–224. url: `https://www.aclweb.org/anthology/P10-2041` (visited on 09/08/2020).

[64] Christopher Olah. *Understanding LSTM Networks*. url: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[65] Sinno Jialin Pan et al. “Cross-domain sentiment classification via spectral feature alignment”. In: *Proceedings of the 19th international conference on World wide web*. 2010, pp. 751–760.

[66] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. url: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[67] Nanyun Peng and Mark Dredze. “Multi-task domain adaptation for sequence tagging”. In: *arXiv preprint arXiv:1608.02689* (2016).

[68] Jeffrey Pennington, Richard Socher, and Christopher D Manning. “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[69] Matthew E Peters et al. “Deep contextualized word representations”. In: *arXiv preprint arXiv:1802.05365* (2018).

[70] Barbara Plank, Anders Johannsen, and Anders Søgaard. “Importance weighting and unsupervised domain adaptation of POS taggers: a negative result”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 968–973.

[71] Barbara Plank and Gertjan van Noord. “Effective Measures of Domain Similarity for Parsing”. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 1566–1576. url: `https://www.aclweb.org/anthology/P11-1157` (visited on 09/08/2020).

[72] Alan Ramponi and Barbara Plank. “Neural Unsupervised Domain Adaptation in NLP—A Survey”. In: *arXiv:2006.00632 [cs]* (May 2020). arXiv: 2006.00632. url: `http://arxiv.org/abs/2006.00632` (visited on 08/28/2020).

[73] Robert Remus. “Domain Adaptation Using Domain Similarity- and Domain Complexity-Based Instance Selection for Cross-Domain Sentiment Analysis”. In: *2012 IEEE 12th International Conference on Data Mining Workshops*. ISSN: 2375-9259. Dec. 2012, pp. 717–723. doi: `10.1109/ICDMW.2012.46`.

[74] Anthony Rios, Ramakanth Kavuluru, and Zhiyong Lu. “Generalizing biomedical relation classification with neural adversarial domain adaptation”. en. In: *Bioinformatics* 34.17 (Sept. 2018). Publisher: Oxford Academic, pp. 2973–2981. issn: 1367-4803. doi: `10.1093/bioinformatics/bty190`. url: `https://academic.oup.com/bioinformatics/article/34/17/2973/4953706` (visited on 08/29/2020).

[75] Gil Rocha and Henrique Lopes Cardoso. “A Comparative Analysis of Unsupervised Language Adaptation Methods”. In: *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*. 2019, pp. 11–21.

[76] Guy Rotman and Roi Reichart. “Deep contextualized self-training for low resource dependency parsing”. In: *Transactions of the Association for Computational Linguistics* 7 (2019), pp. 695–713.

[77] Sebastian Ruder. “An overview of multi-task learning in deep neural networks”. In: *arXiv preprint arXiv:1706.05098* (2017).

[78] Sebastian Ruder and Barbara Plank. “Learning to select data for transfer learning with Bayesian Optimization”. In: *arXiv:1707.05246 [cs]* (July 2017). arXiv: 1707.05246. url: `http://arxiv.org/abs/1707.05246` (visited on 09/08/2020).

[79] Sebastian Ruder and Barbara Plank. “Strong Baselines for Neural Semi-supervised Learning under Domain Shift”. In: *arXiv:1804.09530 [cs, stat]* (Apr. 2018). arXiv: 1804.09530. url: `http://arxiv.org/abs/1804.09530` (visited on 05/24/2019).

[80] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. “Asymmetric Tri-training for Unsupervised Domain Adaptation”. In: *arXiv:1702.08400 [cs]* (Feb. 2017). arXiv: 1702.08400. url: `http://arxiv.org/abs/1702.08400` (visited on 05/24/2019).

[81] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. “Asymmetric tri-training for unsupervised domain adaptation”. In: *arXiv preprint arXiv:1702.08400* (2017).

[82] Motoki Sano et al. “Adversarial training for cross-domain universal dependency parsing”. In: *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. 2017, pp. 71–79.

[83] Motoki Sato et al. “Adversarial Training for Cross-Domain Universal Dependency Parsing”. In: *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 71–79. doi: `10.18653/v1/K17-3007`. url: `https://www.aclweb.org/anthology/K17-3007` (visited on 08/29/2020).

[84] Darsh J. Shah et al. “Adversarial Domain Adaptation for Duplicate Question Detection”. In: *arXiv:1809.02255 [cs]* (Sept. 2018). arXiv: 1809.02255. url: `http://arxiv.org/abs/1809.02255` (visited on 08/29/2020).

[85] Darsh J Shah et al. “Adversarial domain adaptation for duplicate question detection”. In: *arXiv preprint arXiv:1809.02255* (2018).

[86] Jian Shen et al. “Wasserstein distance guided representation learning for domain adaptation”. In: *arXiv preprint arXiv:1707.01217* (2017).

[87] Ge Shi et al. “Genre separation network with adversarial training for cross-genre relation extraction”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 1018–1023.

[88] Rui Shu et al. “A dirt-t approach to unsupervised domain adaptation”. In: *arXiv preprint arXiv:1802.08735* (2018).

[89] Anders Søgaard. “Simple Semi-Supervised Training of Part-Of-Speech Taggers”. In: *Proceedings of the ACL 2010 Conference Short Papers*. Uppsala, Sweden: Association for Computational Linguistics, July 2010, pp. 205–208. url: `https://www.aclweb.org/anthology/P10-2038` (visited on 09/08/2020).

[90] Anders Søgaard. “Simple semi-supervised training of part-of-speech taggers”. In: *Proceedings of the ACL 2010 Conference Short Papers*. 2010, pp. 205–208.

[91] Chi Sun et al. “How to fine-tune bert for text classification?” In: *China National Conference on Chinese Computational Linguistics*. Springer. 2019, pp. 194–206.

[92] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[93] Chengyu Wang et al. “Meta Fine-Tuning Neural Language Models for Multi-Domain Text Mining”. In: *arXiv preprint arXiv:2003.13003* (2020).

[94] Marlies van der Wees, Arianna Bisazza, and Christof Monz. “Dynamic Data Selection for Neural Machine Translation”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 1400–1410. doi: `10.18653/v1/D17-1147`. url: `https://www.aclweb.org/anthology/D17-1147` (visited on 09/08/2020).

[95] Thomas Wolf et al. “HuggingFace’s Transformers: State-of-the-art Natural Language Processing”. In: *ArXiv* abs/1910.03771 (2019).

[96] Yonghui Wu et al. “Google’s neural machine translation system: Bridging the gap between human and machine translation”. In: *arXiv preprint arXiv:1609.08144* (2016).

[97] Hu Xu et al. “BERT Post-Training for Review Reading Comprehension and Aspect-based Sentiment Analysis”. 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, pp. 2324–2335.

[98] Zhilin Yang et al. “Xlnet: Generalized autoregressive pretraining for language understanding”. In: *Advances in neural information processing systems*. 2019, pp. 5753–5763.

[99] David Yarowsky. “Unsupervised Word Sense Disambiguation Rivaling Supervised Methods”. In: *33rd Annual Meeting of the Association for Computational Linguistics*. Cambridge, Massachusetts, USA: Association for Computational Linguistics, June 1995, pp. 189–196. doi: `10.3115/981658.981684`. url: `https://www.aclweb.org/anthology/P95-1026` (visited on 09/04/2020).

[100] Michihiro Yasunaga, Jungo Kasai, and Dragomir Radev. “Robust Multilingual Part-of-Speech Tagging via Adversarial Training”. In: *arXiv:1711.04903 [cs]* (Apr. 2018). arXiv: 1711.04903. url: `http://arxiv.org/abs/1711.04903` (visited on 08/29/2020).

[101] Hai Ye et al. “Feature Adaptation of Pre-Trained Language Models across Languages and Domains for Text Classification”. In: *arXiv:2009.11538 [cs]* (Sept. 2020). arXiv: 2009.11538. url: `http://arxiv.org/abs/2009.11538` (visited on 09/29/2020).

[102] Han Zhao et al. “Adversarial multiple source domain adaptation”. In: *Advances in neural information processing systems*. 2018, pp. 8559–8570.

[103] Zhi-Hua Zhou and Ming Li. “Tri-training: Exploiting unlabeled data using three classifiers”. In: *IEEE Transactions on knowledge and Data Engineering* 17.11 (2005), pp. 1529–1541.

[104] Zhi-Hua Zhou and Ming Li. “Tri-training: exploiting unlabeled data using three classifiers”. In: *IEEE Transactions on Knowledge and Data Engineering* 17.11 (Nov. 2005). Conference Name: IEEE Transactions on Knowledge and Data Engineering, pp. 1529–1541. issn: 1558-2191. doi: `10.1109/TKDE.2005.186`.

[105] Yftah Ziser and Roi Reichart. “Neural Structural Correspondence Learning for Domain Adaptation”. In: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 400–410. doi: `10.18653/v1/K17-1040`. url: `https://www.aclweb.org/anthology/K17-1040` (visited on 05/14/2020).

[106] Yftah Ziser and Roi Reichart. “Pivot Based Language Modeling for Improved Neural Domain Adaptation”. 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, June 2018, pp. 1241–1251. doi: `10.18653/v1/N18-1112`.

[107] Yftah Ziser and Roi Reichart. “Pivot Based Language Modeling for Improved Neural Domain Adaptation”. 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, June 2018, pp. 1241–1251. doi: `10.18653/v1/N18-1112`. url: `https://www.aclweb.org/anthology/N18-1112` (visited on 01/24/2020).

[108] Yftah Ziser and Roi Reichart. “Task Refinement Learning for Improved Accuracy and Stability of Unsupervised Domain Adaptation”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 5895–5906. doi: `10.18653/v1/P19-1591`. url: `https://www.aclweb.org/anthology/P19-1591` (visited on 01/24/2020).