Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Detection of Causality Relations in Plain Text with the use of Word Embeddings

Διπλωματικη Εργασια

του

Μπάστα Γρηγόρη

Εξωτερικός Επιβλέπων: Philippe Muller
Associate Professor UPS

Επιβλέπων Ε.Μ.Π: Ανδρέας-Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2017

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Detection of Causality Relations in Plain Text with the use of Word Embeddings

Διπλωματικη Εργασια

του

Μπάστα Γρηγόρη

Εξωτερικός Επιβλέπων: Philippe Muller
Associate Professor UPS

Επιβλέπων Ε.Μ.Π: Ανδρέας-Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 27η Σεπτεμβρίου 2017.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
....	....	....
Ανδρέας-Γ. Σταφυλοπάτης	Παναγιώτης Τσανάκας	Γεώργιος Στάμου
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2017

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντά μου κ. Philippe MUller, Associate Professor στο Université Toulouse III - Paul Sabatier (υπεύθυνο της ομάδας MELODI-IRIT). Ακόμα, όλα τα μέλη της ομάδας MELODI της οποίας υπήρξα προσωρινό μέλος σε όλη τη διάρκεια της συμμετοχής μου στο πρόγραμμα Erasmus+ μέσω του οποίου μου δόθηκε η δυνατότητα να υλοποιήσω την εργασία μου στην πόλη Toulouse της Γαλλίας. Επιπλέον, οφείλω να ευχαριστήσω τον κ. Andreas Herzig (Directeur de Recherches CNRS) για την έγκριση του να εργαστώ στο IRIT και την καθοδήγησή του τόσο για την επιλογή των συνεργατών μου όσο και του θέματος της διπλωματικής μου.

Επίσης, ευχαριστώ ιδιαίτερα τον επιβλέποντα καθηγητή του Ε.Μ.Π. κ. Ανδρέα-Γεώργιο Σταφυλοπάτη για τη δυνατότητα που μου έδωσε να φέρω εις πέρας τη διπλωματική μου εργασία στο εξωτερικό και για όλες τις χρήσιμες συμβουλές που μου παρείχε.

Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου, Φούλη και Κωστή, για την αμέριστη συμπαράσταση τους στις επιλογές μου, τους φίλους που απέκτησα κατά την παραμονή μου στην Toulouse, τους συναδέλφους μου από το IRIT και τους φίλους μου από την Ελλάδα, που όλοι τους με στήριξαν στις δυσκολίες που αντιμετώπισα.

Περίληψη

Το πρόβλημα με το οποίο καταπιανόμαστε σε αυτή την εργασία είναι αυτό της ανίχνευσης σχέσεων αιτιότητας σε φυσική γλώσσα. Δουλέψαμε συγκεκριμένα πάνω στη γαλλική γλώσσα αξιοποιώντας, όμως, μια μεθοδολογία αρκετά γενική ώστε να μπορεί να αξιοποιηθεί με μικρές προσαρμοστικές ρυθμίσεις και σε πληθώρα άλλων γλωσσών.

Αρχικά θα προσπαθήσουμε να καταδείξουμε τη φύση του προβλήματος αξιοποιώντας και ορισμένα παραδείγματα. Μια χρήσιμη για την εργασία μας διάκριση είναι αυτή μεταξύ ρητής (explicit) και υπόρρητης (implicit) αιτιότητας. Στην πρώτη περίπτωση, η αιτιότητα γίνεται έκδηλη μέσω μίας λεκτικής μονάδας που λειτουργεί ως μη-αμφίσημος δείκτης (indicator) της εν λόγω σχέσης. Στη δεύτερη περίπτωση, η ανίχνευση της αιτιακής σχέσης δεν αποτελεί τετριμμένο πρόβλημα όπως στην πρώτη. Κατά βάση, για να συναγάγουμε πιθανές αιτιακές συνδέσεις, στηριζόμαστε στις σημασιολογικές σχέσεις μεταξύ λέξεων. Μέλημα μας είναι ο υπολογιστής να μπορεί να ανιχνεύει την αιτιότητα και στις δύο περιπτώσεις.

Ένα παράδειγμα στιγμιοτύπου ρητής αιτιότητας είναι: «Υπήρξαν σοβαροί τραυματισμοί εξαιτίας του τυφώνα». Εδώ εμφανίζεται ο μη-αμφίσημος δείκτης αιτιότητας «εξαιτίας». Παραδείγματα στιγμιοτύπων υπόρρητης αιτιότητας είναι: «Είχα πονοκεφάλους από την έκθεση στο καυσαέριο» και «Τα μηχανήματα ήταν παλιά. Ένας εργάτης τραυματίστηκε σοβαρά.» Σε κανένα από τα δύο τελευταία παραδείγματα δεν εμφανίζεται κάποια λεκτική μονάδα που να υποδεικνύει πρόδηλα σχέση αιτίου-αποτελέσματος.

Υπάρχουν δύο βασικές κατευθυντήριες γραμμές για την αντιμετώπιση του προβλήματος που παρουσιάσαμε. Από τη μία πλευρά μπορούμε να βασιστούμε στη χρήση γλωσσολογικών μεθόδων, να αξιοποιήσουμε δηλαδή συντακτικά και σημασιολογικά εργαλεία. Από την άλλη, μπορούμε να βασιστούμε στη χρήση στατιστικών μοντέλων και ευφυών συστημάτων. Εδώ επιλέξαμε να επικεντρωθούμε στην δεύτερη μέθοδο αντιμετώπισης του προβλήματος, αλλά ταυτόχρονα αξιοποιήσαμε και την πρώτη στη φάση της συλλογής των δεδομένων εκπαίδευσης για τα υπολογιστικά μας μοντέλα.

Πιο συγκεκριμένα μπορούμε να χωρίσουμε την εργασία μας σε τρία στάδια. Το πρώτο στάδιο αφορά την εξαγωγή δεδομένων εκπαίδευσης από γαλλικό corpus. Στο δεύτερο στάδιο εκπαιδεύουμε τα μοντέλα μηχανικής μάθησης (Word2vec, SVD, NMF) με στόχο τη δημιουργία Word Embeddings. Στο τρίτο στάδιο αξιολογούμε τα αποτελέσματα μας.

Η μέθοδος μας μπορεί να συνοψιστεί στη λογική της αξιοποίησης στιγμιοτύπων *ρητής* αιτιότητας για την εκπαίδευση μοντέλων παραγωγής πυκνών λεκτικών διανυσματικών αναπαραστάσεων. Στόχος είναι η αναγνώριση αιτιακά συνδεδεμένων λέξεων στη βάση της σημασιολογικής τους αλληλοσυσχέτισης και κατ΄ επέκταση, η ανίχνευση σχέσεων αιτίου-αποτελέσματος σε στιγμιότυπα *υπόρρητης* αιτιότητας. Δηλαδή, από τις ρητές αιτιακές εκφάνσεις «πηδάμε» στις άρρητες. Ουσιαστικά, εκπαιδεύουμε το μοντέλο μας ώστε να προβλέπει την πιθανότητα της (συν-)εμφάνισης σε στιγμιότυπο σχέσης αιτιότητας, μιας λέξης w (είσοδος), με όλες τις άλλες λέξεις του λεξιλογίου μας. Στηριζόμαστε στην υπόθεση ότι η σημασιολογική σχέση μεταξύ δύο λέξεων ή δύο κατηγορημάτων, μπορεί να εξαχθεί (με καλό ποσοστό επιτυχίας) από τον τρόπο που ήδη αυτές/ά συσχετίζονται συντακτικά σε έκδηλα αιτιακό σημασιολογικό πλαίσιο στο παρελθόν.

Η φάση της συλλογής δεδομένων εκπαίδευσης στηρίζεται συγκεκριμένα στην αξιοποίηση συγκεκριμένων συντακτικών μοτίβων που έχουν ως κεντρικό «σπόρο» ένα σύνολο από αιτιακούς δείκτες της επιλογής μας, εξού και η ονομασία seed patterns. Χρησιμοποιούμε αυτά τα μοτίβα για να συλλέξουμε τις δύο διακριτές αιτιακές συνιστώσες που συγκροτούν την αιτιακή σχέση, δηλαδή την αναπαράσταση σε φυσική γλώσσα δύο αιτιακά συζευγμένων γεγονότων, του αίτιου και του αποτελέσματος. Κατά βάση, θέλουμε να αποθηκεύουμε τις λέξεις που συμπυκνώνουν καλύτερα το σημασιολογικό περιεχόμενο του εκάστοτε αιτιακού γεγονότος. Αυτό σημαίνει ότι αγνοούμε ένα σύνολο λέξεων που ονομάζουμε stop-words, λέξεις που εμφανίζονται πολύ συχνά στον καθημερινό λόγο και δεν συμπυκνώνουν πολύ πληροφορία.

Τα δεδομένα μας συλλέγονται από το frWac (annotated) σώμα κειμένου και αποθηκεύονται σε αυστηρά δομημένη και πλούσια σε πληροφορία μορφή σε ένα αρχείο xml. Αυτό το αρχείο αξιοποιείται για την εξαγωγή δεδομένων εκπαίδευσης σε σχετικά σύντομο χρόνο και για πολλές διαφορετικές παραμέτρους. Θα εξετάσουμε σύντομα τι είναι αυτές οι παράμετροι. Η μείωση του χρόνου είναι σημαντική μιας και στο xml αρχείο μας έχουμε αποθηκευμένες μοναχά προτάσεις που αποτελούν ρητά αιτιακά στιγμιότυπα και δεν χρειάζεται πλέον να προσπελαύνουμε όλο το σώμα κειμένου σε αναζήτηση για αιτιακούς δείκτες. Αναζητούμε κάθε μία ξεχωριστά τις λέξεις κάθε πρότασης του σώματος κειμένου και στην περίπτωση που πετυχαίνουμε έναν πρόδηλο αιτιακό δείκτη (π.χ. à_cause_de) κρατάμε την τρέχουσα πρόταση και συλλέγουμε τα χρήσιμα αιτιακά δεδομένα. Τα αιτιακά δεδομένα είναι οι επιμέρους λέξεις των αιτιακών συνιστωσών μαζί με άλλες μορφολογικές, γραμματικές και συντακτικές πληροφορίες για κάθε μία εξ αυτών.

Ας εξετάσουμε όμως σε μεγαλύτερο βάθος αυτό που αποκαλέσαμε seed patterns. Πρόκειται ουσιαστικά για ένα σύνολο κανόνων βασισμένων σε λογικές συνθήκες που αφορούν συντακτικές και γραμματικές σχέσεις. Βασικός οδηγός μας αποτελεί το Δένδρο Εξαρτήσεων (Dependency Tree) που αποτυπώνει τις συντακτικές σχέσεις μεταξύ όλων των λέξεων κάθε πρότασης. Όταν βρίσκουμε έναν αιτιακό δείκτη τότε ακολουθώντας τις λογικές συνθήκες που υποδεικνύουν τα μοτίβα μας, αποθηκεύουμε ένα υποσύνολο των λέξεων του Δένδρου Εξαρτήσεων κάνοντας ουσιαστικά αναζήτηση πάνω στους κόμβους-λέξεις του, εκκινώντας από

τον κόμβο-ρίζα του αιτιακού δείκτη και κάνοντας διάκριση μεταξύ του υποδένδρου με ρίζα τον πρότερο γονέα και του υποδένδρου με ρίζα τα πρότερα παιδιά του κόμβου (που πλέον αντιμετωπίζουμε ως ρίζα). Η διάκριση αυτή καθορίζει το σημασιολογικό διαχωρισμό των αιτιακών γεγονότων σε αίτια και αποτελέσματα.

Αξιοποιούμε τρία βασικά μοτίβα που διαφέρουν αρχικά ως προς το μέρους του λόγου του αιτιακού δείκτη (πρόθεση, σύνδεσμος, ρήμα). Στην περίπτωση του συνδέσμου (π.χ. car) θεωρούμε ότι οι αιτιακές προτάσεις στην πλειοψηφία τους δομούνται στη μορφή: Ρηματικό Σύνολο (αποτέλεσμα) – Σύνδεσμος – Ρηματικό Σύνολο (αίτιο). Αντίστοιχα για τις προθέσεις (π.χ. grâce_à) : Ρηματικό Σύνολο (αποτέλεσμα) – Πρόθεση – Ονοματικό Σύνολο (αίτιο). Τέλος για τους δείκτες-ρήματα (π.χ. causer) προκύπτουν δύο διαφορετικά (υπό-)μοτίβα ένα για την περίπτωση που το ρήμα εμφανίζεται σε ενεργητική φωνή και ένα όταν εμφανίζεται σε παθητική. Ενεργητική φωνή: Ονοματικό Σύνολο (αίτιο) - Ρήμα – Ονοματικό Σύνολο (αποτέλεσμα). Παθητική φωνή: Ονοματικό Σύνολο (αποτέλεσμα) – Ρήμα – Ονοματικό Σύνολο (αίτιο). Τα μοτίβα μας είναι λοιπόν τέτοια ώστε να συλλαμβάνουν αυτές τις διαφορετικές συντακτικές μορφές εμφάνισης των αιτιακών σχέσεων. Το σύνολο των αιτιακών δεικτών που χρησιμοποιήθηκαν συλλέχθηκαν από το Asfalda French FrameNet.

Οι εξαγωγές δεδομένων σε κάθε περίπτωση μπορούν να οδηγήσουν σε αποθήκευση πολύ μεγάλου αριθμού λέξεων. Στην πραγματικότητα, όμως, σπάνια αξιοποιούμε όλο τον όγκο αυτής της πληροφορίας. Μάλιστα πολύ συχνά ολόκληρες αιτιακές προτάσεις απορρίπτονται τελείως γιατί δεν έχουν καμία χρήσιμη πληροφορία να μας δώσουν, δηλαδή δεν περιλαμβάνουν non-stopwords που να συμπυκνώνουν το νόημα των αιτιακών γεγονότων. Ακόμα, κάποιες προτάσεις στιγμιότυπα αιτιακών σχέσεων δεν ταίριαζαν στα μοτίβα εξαγωγής δεδομένων είτε λόγω θορύβου στο αρχικό αδόμητο σώμα κειμένου και λαθών στη φάση του annotation, είτε λόγω προτάσεων που απλούστατα αποτελούν εξαίρεση στις δομικές μορφές εμφάνισης ρητής αιτιότητας που παρουσιάσαμε.

Την «ευθύνη» της αγνόησης ορισμένων δεδομένων του xml αρχείου αναλαμβάνει ένα επιπλέον σύνολο από λογικές συνθήκες που λειτουργεί ουσιαστικά ως φίλτρο θορύβου και περεταίρω πληροφορίας, που ανάλογα με τις απαιτήσεις μας για κάθε περίπτωση δοκιμαστικού συνόλου εκπαίδευσης, θεωρούμε πλεονάζουσες. Είναι ακριβώς αυτό το φίλτρο που αναλαμβάνει να απορρίψει όλες τις non-stopwords λέξεις που δεν ανήκουν σε ένα πλούσιο γαλλικό λεξικό, λέξεις που δεν ξεκινάνε από αλφαριθμητικό χαρακτήρα ή από πεζό γράμμα (έτσι απορρίπτουμε αριθμούς και κύρια ονόματα). Ακόμα, το φίλτρο αυτό περιλαμβάνει κάποιες λογικές συνθήκες που συνιστούν ρυθμιζόμενες παραμέτρους του γιατί μπορούν να αλλάζουν ανάλογα με τα επιθυμητά χαρακτηριστικά του συνόλου εκπαίδευσης μας. Οι παράμετροι αυτοί αφορούν τη ρύθμιση του βάθους αναζήτησης στο Δένδρο Εξαρτήσεων μας και την επιλεκτική χρήση μονάχα κάποιων εκ των αιτιακών δεικτών, είτε ανάλογα με τα πλαίσια (του FrameNet) στα οποία ανήκουν, είτε με το μέρος του λόγου τους. Με τη ρύθμιση αυτών των παραμέτρων μπορούμε να δημιουργήσουμε δεδομένα εκπαίδευσης που είτε χαρακτηρίζονται από την έμφαση στα ποιοτικά είτε στα ποσοτικά χαρακτηριστικά. Να τονίσουμε σε αυτό το σημείο ότι

κατά τη φάση δημιουργίας των δεδομένων εκπαίδευσης από το xml αρχείο χρησιμοποιούμε μονάχα λήμματα και έτσι αναφερόμενοι προηγουμένως σε «λέξεις» ουσιαστικά εννοούσαμε «λήμματα».

Ύστερα, περνάμε στο στάδιο της εκπαίδευσης των μοντέλων μας. Πρώτα όμως, ας εξετάσουμε τις διαφορές ως προς τη δομή και τη λειτουργία των τελευταίων. Το Word2Vec είναι μία οικογένεια μοντέλων (νευρωνικών δικτύων) που δέχονται ως είσοδο επιμέρους λέξεις και εκπαιδεύονται ώστε να προβλέπουν την πιθανότητα συν-εμφάνισης αυτών των λέξεων με όλες τις άλλες που έχουμε στο λεξιλόγιό μας. Αυτό ακριβώς το χαρακτηριστικό του Word2Vec είναι που εκμεταλλευόμαστε για να δημιουργήσουμε τέτοιες διανυσματικές αναπαραστάσεις ώστε να συμπυκνώνουν πληροφορία για την πιθανότητα συν-εμφάνισης δύο λέξεων ως στοιχεία αιτιακών προτάσεων. Το καθοριστικότερο συστατικό στοιχείο του Word2vec είναι ένα επίπεδο που υλοποιεί softmax regression. Οι παράμετροι της softmax συνάρτησης είναι οι ίδιες οι διανυσματικές αναπαραστάσεις των λέξεων του λεξιλογίου που χρησιμοποιούμε. Ποιο συγκεκριμένα δημιουργούνται δύο ξεχωριστοί πίνακες διανυσμάτων, ένας εισόδου και ένας εξόδου. Στην πράξη αξιοποιείται μία προσέγγιση της softmax συνάρτησης, η μέθοδος Negative Sampling. Η υποθετική συνάρτηση αυτού του μοντέλου μας υποδεικνύει ότι δύο διανύσματα έχουν μικρό εσωτερικό γινόμενο όταν έχουν μεγάλη πιθανότητα συν-εμφάνισης. Μια παραλλαγμένη εκδοχή αυτού του μέτρου (δηλ. του εσωτερικού γινομένου) χρησιμοποιούμε και για να μετρήσουμε το βαθμό αιτιακής συσχέτισης δύο λέξεων, κάτι που ονομάζουμε αιτιακή εγγύτητα (causal proximity). Τα επόμενα μοντέλα που χρησιμοποιήσαμε στηρίζονται στην παραγοντοποίηση πίνακα με δύο διαφορετικές μεθόδους. Η πρώτη είναι η Singular Value Decomposition (SVD) ενώ η δεύτερη είναι η Non-negative Matrix Factorization.

Η πρώτη μορφή δεδομένων εισόδου που αξιοποιήθηκε για την εκπαίδευση των μοντέλων μας είναι κοινή και για τα τρία. Δημιουργήσαμε δεδομένα εκπαίδευσης στη μορφή ζευγών λέξεων που συγκροτούν το αίτιο (cause-words) και λέξεων που συγκροτούν το αποτέλεσμα (effect-word) αξιοποιώντας το xml αρχείο μας και το φίλτρο που περιγράψαμε. Κάθε cause-word έχει για context μία effect-word και αντίστροφα.

Το Word2vec προσλαμβάνει αυτά τα ζεύγη ως λέξεις που η παρουσία της μίας (είσοδος) πρέπει να προβλέπει την παρουσία της άλλης (έξοδος). Η λέξη «παρουσία» εν προκειμένω αφορά την αιτιακή εγγύτητα δύο λέξεων. Θεωρούμε για τους λόγους που εξηγήθηκαν πιο πάνω ότι αυτή η πληροφορία ενσωματώνεται στον εξωτερικό και εσωτερικό πίνακα των διανυσματικών μας αναπαραστάσεων και πιο συγκεκριμένα στη σχέση αυτών. Αυτό το σκεπτικό διαφοροποιείται σημαντικά από τον τρόπο που χρησιμοποιείται συνήθως το μοντέλο Word2vec, όπου αξιοποιείται μονάχα ο πίνακας εισόδου. Οδηγούμαστε στη συγκεκριμένη στρατηγική αξιοποίησης του εν λόγω μοντέλου καθώς δεν το αξιοποιούμε για να ανιχνεύσουμε σχέσεις ομοιότητας όπως είθισται, αλλά σχέσεις αιτιότητας. Το συγκεκριμένο μοντέλο το ονομάσαμε Single Word-pair Model σε αντιδιαστολή με την εναλλακτική μέθοδο εκπαίδευσης του Word2vec που θα παρουσιάσουμε παρακάτω.

Έπειτα, κατά τη χρήση των μοντέλων παραγοντοποίησης πίνακα επιστρατεύσαμε τη συνάρτηση PMI (Pointwise Mutual Information) για να δημιουργήσουμε τον αρχικό μας πίνακα. Οι γραμμές αντιστοιχούν στα cause-words και οι στήλες στα effect-words. Κάθε κελί περιέχει την τιμή PMI των λέξεων της αντίστοιχης στήλης και γραμμής. Θεωρούμε ότι η συνάρτηση PMI είναι ένα πολύ αντιπροσωπευτικό μέτρο για την αιτιακή εγγύτητα. Ο λόγος που δεν χρησιμοποιούμε απευθείας το PMI αλλά επιλέξαμε την παραγοντοποιήση του εν λόγω πίνακα είναι ότι υπάρχουν πολλά ζευγάρια λέξεων που δεν συν-εμφανίζονται ποτέ στα αιτιακά στιγμιότυπα που εξαγάγαμε από το σώμα κειμένου μας. Ακόμα, η παραγοντοποίηση πίνακα δίνει τη δυνατότητα υψηλής γενίκευσης στις προβλέψεις μας.

Χρησιμοποιήθηκε μία ακόμη εκπαιδευτική μέθοδος η οποία διαφοροποιείται από τις προηγούμενες ως προς τη μορφή των δεδομένων εισόδου. Αυτή τη φορά αξιοποιήσαμε μόνο το Word2vec, όμως, αντί να του παρουσιάσουμε ως ξεχωριστές προτάσεις απλά ζευγάρια cause-words και effect-words, του δίνουμε όλες τις λέξεις των αιτιακών συνιστωσών κάθε πρότασης στη σειρά. Έτσι, ως πλαίσιο μιας λέξης μπορεί να θεωρηθούν όλες οι υπόλοιπες λέξεις των αιτιακών συνιστωσών πλην της ίδιας. Για κάθε λέξη προσπαθούμε να προβλέψουμε το πλαίσιό της. Με την ίδια λογική όπως και προηγουμένως αξιοποιούμε τα embeddings εισόδου και εξόδου για τα test που ακολουθούν. Η εν λόγω μέθοδος παρουσιάζει την παραδοξότητα ότι πλεον θα προβλέπεται μεγάλη εγγύτητα όχι μόνο μεταξύ διανυσμάτων λέξεων από διαφορετικά αιτιακά γεγονότα, αλλά και λέξεων του ίδιου γεγονότος. Αυτό δεν μας εμποδίζει παρ' όλα αυτά να προβλέπουμε αιτιακή εγγύτητα, δεδομένου ότι την εξετάζουμε μονάχα για λέξεις που ξέρουμε (ή τουλάχιστον υποθέτουμε) ότι υποδηλώνουν διαφορετικά γεγονότα. Αυτή η τελευταία συνθήκη αποτελεί προϋπόθεση για την καλή λειτουργία όλων των μοντέλων μας και αποτελεί καλώς ή κακώς έναν περιορισμό στο στόχο μας για την ανίχνευση σχέσεων αιτιότητας. Η τελευταία αυτή μέθοδος αποτελεί πάντως με εμφανή διαφορά την αποτελεσματικότερη από όλες τις μεθόδους που παρουσιάστηκαν έως τώρα. Το γεγονός αυτό οφείλεται στο μεγάλο βαθμό γενίκευσης που επιτυγχάνει. Λόγω του ότι στηρίζεται στην εκαπίδευση πάνω σε ζεύγη αιτιακών συνιστωσών (tuples) και όχι σε ζεύγη λέξεων, ονομάζουμε αυτό το μοντέλο μας Tuple-based model.

Η αξιολόγηση των μοντέλων μας έγινε πάνω σε ζεύγη γαλλικών λέξεων του Task8 του Semeval. Πρόκειται, πιο συγκεκριμένα, για ζεύγη ουσιαστικών που έχουν επιλεγεί με τέτοιο τρόπο ώστε να συμπυκνώνουν κατά το δυνατόν πιο αντιπροσωπευτικά τη σχέση της οποίας οι αντίστοιχες προτάσεις αποτελούν στιγμιότυπο. Δεν έχουμε να αντιμετωπίσουμε, λοιπόν, μόνο αιτιακά στιγμιότυπα αλλά πολλές ακόμα σχέσεις. Πρέπει, εν τέλει, να μπορέσουμε να προβλέψουμε με το μεγαλύτερο δυνατό ποσοστό επιτυχίας αν κάθε ένα από τα ζεύγη του Task8 είναι ή όχι αιτιακό στιγμιότυπο.

Το μέτρο που χρησιμοποιήσαμε για τις προβλέψεις μας είναι το cosine similarity. Πρόκειται για μία κανονικοποιημένη μορφή του εσωτερικού γινομένου μεταξύ των διανυσματικών αναπαραστάσεων λέξεων διαφορετικής αιτιακής συνιστώσας. Όσον αφορά το Word2vec, πάντα αξιοποιούμε τα διανύσματα του εσωτερικού (1ο επίπεδο) και εξωτερικού πίνακα (2ο επίπεδο). Αντίστοιχα για τις μεθόδους παραγοντοποίησης πίνακα αξιοποιούμε τις διανυσματικές αναπαραστάσεις που αποτυπώνονται στους δύο πίνακες που προκύπτουν.

Εξετάσαμε τη συμπεριφορά των μοντέλων μας χρησιμοποιώντας την γραφική αναπαράσταση Precision-Recall και Receiver Operating Characteristic. Την καλύτερη συμπεριφορά επιδεικνύει το Tuple-based μοντέλο μας με AUC (Area Under Curve) ίσο με 0,69 για το Precision-Recall. Έπειτα ακολουθούν το SVD (AUC = 0,63), το Single Word-pair (AUC = 0,61) και το NMF (AUC = 0,58), με αυτή τη σειρά. Χρησιμοποιήσαμε άκομα ένα μοντέλο ως baseline, συγκεκριμένα ένα Word2Vec μοντέλο προ-εκπαιδευμένο πάνω στο σώμα κειμένου frWac (το ίδιο δηλαδή που χρησιμοποιήσαμε κι εμείς για την εξαγωγή των αιτιακών μας δεδομένων). Όπως ήταν αναμενόμενο, το εν λόγω μοντέλο δεν επέδειξε καλή συμπεριφορά στην αναγνώριση αιτιακά συσχετιζόμενων ζευγών λέξεων (AUC = 0,58). Η επιλογή των παραμέτρων κάθε μοντέλου μας προέκυψε μετά από πολλές δοκιμές διαφορετικών δεδομένων εκπαίδευσης (διαφορετικές ρυθμίσεις στο φίλτρο μας) και δοκιμές για διαφορετικά χαρακτηριστικά του μοντέλου καθ' αυτού. Έχουμε καταγράψει σε γραφικές παραστάσεις το πως ακριβώς επηρεάζουν τέτοιες ρυθμίσεις συγκεκριμένο το Tuple-based μοντέλο μας. Ακόμα αναπαριστούμε τη συμπεριφορά μίας αμφίδρομης (bi-directional) παραλλαγής του SVD μοντέλου μας.

Τέλος, καταλήξαμε σε κάποια γενικότερα συμπεράσματα. Μετά από τη σύγκριση της συμπεριφοράς των μοντέλων μας και την κατανόηση της εσωτερικής δομής και λειτουργίας τους έγινε εμφανής η σημασία της ικανότητας του μοντέλου για υψηλή γενίκευση. Αυτό φάνηκε κα από την αδυναμία του PMI να καλύψει τις ανάγκες της εργασίας μας καθώς επίσης και από την σαφώς πιο επιτυχημένη λειτουργία του Tuple-based μοντέλου μας σε σχέση με τα υπόλοιπα. Είναι σημαντικό να τονιστεί ότι τα μοντέλα μας αντιμετωπίζουν σχετική δυσκολία στην ανίχνευση αιτιότητας σε πολύπλοκες προτάσεις καθώς πολλές από τις λέξεις που συγκροτούν το νόημά τους αντιμετωπίστηκαν απλά ως stop-words. Θεωρούμε ότι πολλές από τις αστοχίες των μοντέλων μας σχετίζονται με το είδος των δεδομένων εκπαίδευσης. Η επιλογή των κατάλληλων δεδομένων εκπαίδευσης είναι ένα σημαντικό ζήτημα το οποία μάλιστα απαιτεί και διαφορετική αντιμετώπιση για κάθε γλώσσα. Θεωρούμε ότι υπάρχουν πολλά περιθώρια βελτίωσης στη διαδικασία συλλογής αυτών των δεδομένων. Βελτιώσεις σε αυτό το κομμάτι της εργασίας μας θα μπορούσαν να προσδώσουν στα μοντέλα μας μεγαλύτερη αποτελεσματικότητα.

Abstract

Causality detection is one of the most challenging topics in NLP. In this project we tried to cope with this open problem by employing training methods focused on the creation of vector representations of french words. While we only worked on the problem of causality detection in the French language, our methodology is applicable in many other cases thanks to its generality. Our whole project can be separated into three major tasks.

The first task pertains to the creation of our training data through the automatic extraction of cause-effect tuples from a syntactically annotated French corpus. For this purpose, we collected non-ambiguous lexical units from the ASFALDA French FrameNet, that denote causality relations. We, therefore, extracted tuples of meaningful sets of words that represent either the cause or the effect of the captured frame. To achieve all of this, we took advantage of the dependency tree of each sentence and the part-of-speech tag of each word.

The second task deals with the computational processing of our training data extracted in the previous task, in order to create causal word embeddings based on cause-effect context similarity. At this stage, the cause-effect tuples created in the first task are treated in an innovative manner as the training data set for the models Word2vec, SVD and NMF, in such a way as to create causal embeddings.

The third task is about the evaluation of our models. We compared the causal proximity of cause-effect word pairs by comparing the dot product and cosine similarity of the embeddings stored in the input matrix and the embeddings stored in the output matrix of our models. For the evaluation, we use the SemEval Task8 test data (partially translated in French).

Contents

Chapter 1

Introduction

Human readers have the extraordinary capability to infer event causality from plain text. This attribute is partly due to our inclination, as human beings, to easily recognize special lexical units such as because, as a consequence, hence, cause, result, originate, etc., that explicitly indicate such a relation between two events. In this particular type of phrases, causality can be inferred even in the case that we don't have any knowledge of the meaning of the words describing the causal events.

However, another cognitive characteristic that makes us humans capable of detecting causality even in cases where there are no explicit causal lexical units, is our ability to employ semantical relations, intuitional notions about how probable it is that two events are causally connected. Many causality instances in natural language, either involve only ambiguous connectives (*and, from)* or they don't involve any connectives at all. Yet, we are still able to notice if there is a causal link between events merely because we have some prior knowledge that forms our intuitional interpretation abilities.

Here are some examples of event causality instances. The first one is an explicit case of causality where causality is indicated by the noun *cause*:

Suicide is one of the leading ***causes*** *of death.*

The second one is an implicit case involving an ambiguous connective:

He had chest pains and headaches ***from*** *mold.*

The next one does not involve any cause-effect lexical units or any other connectives. The events that are causally linked appear in two separate sentences.

The woman had an infection. She took antibiotics.

The causality relation is more than evident in all of the above cases.

We should also notice that humans have the ability to deny the existence of a causal link. For example, it would be easy for a human to deny a causal relation in the next example of sentence pair that appears in the same form as the sentences in the previous case:

The woman had an infection. She washed her hands.

Even so there are cases where even humans are not sure but they can still can guess quite well and even this attribute is very important for language understanding and communication.

Yet, it is not at all evident how a computer could perform such a task. Thus, there arises a very interesting problem for the NLP community to tackle. There is already a lot of effort given from researchers to cope with such a task, a tricky task, considering the various forms that causality may appear.

In this project our work deals with causality detection specifically in **french** language. We contribute to the relevant research field by making some suggestions about prediction and distributional models that could be employed for direct detection of such relation instances or that could be used as auxiliary tools for future projects that will concentrate more into formal semantic and pragmatic approaches. We also try to make some useful observations derived as general conclusions from our work and specifically from its evaluation part. This commenting gives additional insight into the employed models (word2vec, matrix factorization), their special features and innovative ways that could be used. We also provide the NLP community with a succinct set of data comprising causally connected words extracted from the frWac corpus, and we store them encoded in an .xml file for further use. For each and every word there are stored several grammatical and syntactical information. We consider this dataset a very useful tool for researchers that want to get involved with information retrieval tasks related to causality, specifically in the french language.

In this project, we rallied four main different methods dealing with causality detection. All of them relying on knowledge based on focused distributional similarity, specifically what we call **causal proximity** between individual words. As indicator of the existence of causal relation, we consider the probability of the co-occurrence of individual words related as cause-effect pairs. The first two of our methods rely on the state-of-the-art Word2vec tools designed from Mikolov et al. We trained this model aiming at the creation of word embeddings that bear information useful to deduce causal relations, using two different training techniques. The next two methods that we employed use matrix factorization algorithms. We decided to test the Non-negative Matrix Factorization (NMF) and Singular Value Decomposition (SVD) methods for linear dimensionality reduction of a matrix by factorizing it in matrices that in a way constitute our alternatively trained causal embeddings.

Chapter 2

Semantic Relation Extraction

One of the most important chalenges for the NLP community, today, is the automatic extraction of valid knowledge from plain text. If we want to effectively simulate or at least mimic the humans' ability to understand written texts, we need to develop models that will be able to cope with the various forms of syntax, semantics, a continually evolving vocabulary, and ambiguous linguistic constructs like figurative expressions, metaphors, rhetorics, sarcasm and slang. Many simple tasks like the identification of negation are still problems that need to be solved [1]. We can't neglect, however, the huge breakthroughs made in the field of NLP during the last three decades. The current research seems to progress quite efficiently, especially in what pertains to information extraction tasks.

A critically important research topic that emerged in the past few years is the automated extraction of semantic relations or the practically equivalent topic of semantic relation detection. Relevant research finds great applications in question answering, information retrieval, event prediction, generating future scenarios and decision processing. Typical relations that have raised the interest of NLP researchers are part-whole, if-then, cause-effect. The cause-effect relation, with which we are dealing in this project is strongly connected to decision making and thus it plays a crucial role in human cognition [1].

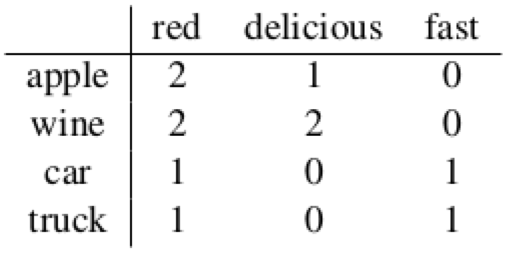
2.1 Methods of semantic relations extraction

Most research on automatic extraction of semantic relations focuses on exploiting large amounts of unannotated corpora, which have become increasingly available for many languages and domains, often by harvesting from the web. Such approaches are based on the distributional hypothesis of Harris [16], stating that words in similar contexts have similar meanings, hence, word meanings can be derived in part from their distribution across different linguistic envionments. Other formulations of the same assertion are the following: “You shall know a word by the company it keeps” as stated by Firth [11] , and also suggested by Harris [17]: “the linguistic meanings which the structure carries can only be due to the relations in which the elements of the structure take part.”

Distributional semantics are more or less founded in this very idea, where finding the meaning of a word is based on its linguistic environment and relevant distributional patterns. This information can be encoded in vectors, one for each word of our vocabulary, that contains the measure of the frequency of co-occurrence of the corresponding word with the rest of our vocabulary. We are going to focus, firstly, on two types of traditional “distributional” approaches that are based on the distributional hypothesis. The first type (distributional approaches) exploits distributional semantic representations indiscriminately in order to infer semantic relatedness between words. The second type of approaches (pattern-based approaches) does the same but exploits only specific types of explicit relations.

2.1.1 Distributional approaches

A great advantage of distributional approaches is that they are purely unsupervised. Vector representations created automatically from large corpora by recording the frequencies of co-occurrence between what we can call target words and context words and by this we mean the surrounding words. We can actually measure the co-occurrence either by using a word context window of fixed size or by exploiting syntactical dependencies. In Figure 2.1, we show examples of co-occurrence counts for several words occurring with different context words.



	red	delicious	fast
apple	2	1	0
wine	2	2	0
car	1	0	1
truck	1	0	1

Figure 2.1: Examples of co-occurence counts for a few english words

Words that have many common co-occurring words are thought to be semantically related, take under consideration the distributional hypothesis (e.g., car and truck have identical vectors in our very simple representation in Figure 2.1, while car and wine have very different vectors). A good measure to represent the proximity of two word vectors considering the amount of common co-occurring words is the cosine similarity.

Budanitsky and Hirst [7] made a clear distinction between semantic similarity and semantic relatedness. They considered the former to be a subset of the latter. Semantic similarity denotes relations of synonymy, hyponymy (and hypernymy), antonymy, or troponymy, while semantic relatedness denotes any semantic relation existing between two words. Furthermore, they formalize this distinction according to the syntactic relations between each word and its co-occurring words context by claiming that in order for two words to be distributionally similar it is necessary to have the same syntactic relation with their co-occurring words and if the don't, they are merely distributionally related. Thus, a way to measure similarity instead of general relatedness is to look specifically at context words that are syntactically related with the target words, instead of using a fixed window of surrounding words.

An important limiting factor of distributional approaches is that distributional measures put some barriers in our will for an effective distinction between different semantic relations, they only achieve a quantification of the level of relatedness between lexical items.

To get at more precise semantic relations, another type of approaches, based on lexico-syntactic patterns, has been investigated by a number of researchers. We present these techniques in the following section.

2.1.2 Pattern-based approaches

Pattern-based approaches are based on a different view of the distributional hypothesis. In distributional approaches, when we measure the relatedness between two words considering their co-occurrence in similar contexts, there is no restriction on the type of contexts. On the contrary, pattern-based approaches target specific relations indicated by explicit lexical units.

Here, we exploit word pairs that are syntactically linked within patterns marked by specific indicators of the targeted relations. Approaches like these are considered to be "weakly supervised" because manual extraction is needed for the specification of patterns [10].

Pattern-based semantic relation extraction usually consists of four main steps: (A) definition of the semantic relation of interest, (B) discovering of the specific patterns which explicitly express these relations and also the syntactic conditions under which the meaning of the targeted relation is realized, (C) the search for instances of the relation by exploiting the patterns, and (D) structuring the new instances as part of a new or existing ontology (or terminological database) [10].

Pattern-based approaches have been shown to achieve high precision and to allow for the identification of particular relations and their distinction. Yet, they tend to have very low recall scores.

2.1.3 Latent Feature Approaches

Latent feature based methods exploit linguistic features extracted from large corpora. For our task we chose to employ and compare Word2vec [25] and Matrix Factorization techniques that create dense vectors, latent feature representations of our lexical units. An important issue that constitutes a prominent research topic is the specific manner that we can use such models not just for semantic similarity tasks but also for tasks of (causality) relation detection. We claim to have some answers on this question, but we are going to discuss more about the ways that word embedding models can be used in the next chapters.

For now, to make a connection between these methods and the previous discussion, we can say that our strategies for relation detection rely on pattern-based approaches either explicitly as with SVD and NMF or implicitly as with Word2vec. In the explicit pattern-based methods we firstly count co-occurrences of words in our carefully chosen causal patterns and we then store these counts in a huge matrix that is then factorized. In the implicit methods we create dense vector representations by training our models to predict a word from its context or inversely. The context of a word is determined by our causal patterns. In all of these models the most distinctive element is the use of latent features, the employment of dense vector representations, what we call word Embeddings.

2.2 Causality detection

The world can be seen as a network of causality where people, organizations, and other kinds of entities causally depend on each other. This network is so huge and complex that it was unavoidable for it to be put under the microscope of science. Hence, causality has been studied extensively in a wide range of disciplines, including Psychology, Linguistics, Philosophy and Computer Science. One of the simplest ways to express cause-effect relations is through propositions of the form 'A causes B' or 'A is caused by B'. It is a highly intuitive notion and yet, the topic has been surrounded by much controversy because experts belonging to these fields often disagree about when two events are causally linked. This is understandable, because causality can be expressed using many different types of propositions (e.g., active, passive, subject-object, nominal or verbal) and have several diverse syntactic representations.

Causality can be expressed using many different types of propositions (e.g., active, passive, subject-object, nominal or verbal) and have several diverse syntactic representations. One popular classification of its explicit representations was given by Khoo et al [22].

1) Causal links can be used to connect clauses or sentences. Altenberg classified causal links into four types: a) adverbial links, e.g. so, hence, therefore, b) prepositional links, e.g. because of, on account of, c) subordination, e.g. because, as, since, and d) clause-integrated links, e.g. that's why, the result was.

2)Causative verbs are transitive verbs whose meanings include a causal element. Examples include break and kill, whose transitive forms are: to cause, to break and to cause to die.

3)Resultative constructions are sentences in which the object of a verb is followed by a phrase describing the state of the object as a result of the action denoted by the verb.An example is 'I painted the car red'.

4)If-Then conditionals often indicate that the antecedent causes the consequent.

5) Causation adverbs and adjectives have causal element in their meanings, e.g. fatal or fatally, that can be paraphrased as to cause to die.

We can discriminate between two basic categories of causal relation detection methods [1]:
I) linguistic, syntactic and semantic pattern matching
II) statistical and machine learning techniques

2.2.1 Statistical vs. Non-Statistical Techniques

Many previous studies have attempted to extract implicit cause-effect relations from text using knowledge-based inferences. These studies were based on hand-coded, domain-specific knowledge bases difficult to scale up for realistic applications. More recently, other researchers (Garcia [12] and Khoo et al. [21]) used linguistic patterns to identify explicit causation relations in text without any knowledge-based inference.

Garcia used French texts to capture causation relationships through linguistic indicators organized in a semantic model which classifies causative verbal patterns. Khoo at al. used predefined verbal linguistic patterns to extract cause-effect information from business and medical newspaper texts. They presented a simple computational method based on a set of partially parsed linguistic patterns that usually indicate the presence of a causal relationship. The relationships were determined by exact matching on text.

The need to make use of a large amount of labelled, domain-and-type-independent, textual data and to extract implicit patterns in text automatically, meant that machine learning techniques could potentially do much better than purely linguistic techniques. Thus, beginning in the early 2000s, the paradigm to tackle the problem of automatic causal relation extraction began shifting to statistics and machine learning. The early studies relied on finding explicitly marked cause-effect pairs in sentence, but with the passage of time, researchers progressively began to account for implicit and ambiguous constructs through careful feature extraction.

Girju [13] was the first one who used machine learning techniques. She specifically employed a supervised method using C4.5 decision trees. A training corpus of 6000 sentences and a test corpus of 1200 sentences containing each of the 60 simple causative verbs was created using a domain-independent text collection. Using a syntactic parser, 6523 relations of the form NP1-Verb-NP2 were found, from which 2101 were causal relations and 4422 were not. These were the positive and negative examples used to train the decision-tree classifier. As features, the constraints on the nouns and verb, which were necessary for a pattern to be a causal relation, were identified. In particular, for each value of NP1 (and similarly for each NP2), nine noun hierarchies from WordNet were used as semantic features: entity, psychological feature, abstraction, state, event, act, group, possession and phenomenon. The training process produced several constraints, which were ranked based on frequency and accuracy.

For our project, other highly influential works in (causality) relation detection are the following.

Juliette Conrath [10] addressed the challenge of relation extraction using a purely distributional method to automatically extract the necessary semantic information for common-sense inference. Typical associations between pairs of predicates and a targeted set of semantic relations (causal, temporal, similarity, opposition, part/whole) were extracted from large corpora, by exploiting the presence of discourse connectives which typically signal these semantic relations.

Sharp et al. [31] generated causal embeddings cost-effectively by bootstrapping cause-effect pairs extracted from free text using a small set of seed patterns. Nextly, they trained dedicated embeddings over these data, by using task-specific contexts, i.e., the context of a cause is its effect. Finally, they extended a state-of-the-art re-ranking approach for QA to incorporate these causal embeddings. For the embeddings creation task they used a dependency-based variant of Mikolov's word2vecf model [25] introduced by Levy and Goldberg [23]. Their model was trained using single word training pairs. They compared several variations of this method and a Convolutional Neural Network, an Alignment model and simple baselines, using test word-pairs drawn from SemEval 2010 Task 8 and by representing their detection capacity using a Precision-Recall curve.

Chapter 3

Machine Learning: Theory and Models

Machine learning is a branch of Artificial Intelligence used for data analysis. It is based on the idea that computers should be able to learn, adapt and thus find structure, predict, cluster and classify data. In this section we are going to present the basic theoretical foundations of machine learning and explain both mathematically and in terms of applicability the functioning of several models either those used in our project for causality detection or other ones that are strongly connected with them.

3.1 Supervised learning

A supervised learning problem is, given a training set $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)}))$, to learn a function h : X $\rightarrow$ Y so that h(x) can effectively predict the corresponding value of y [27]. X and Y denote the space of input values and the space of output values respectively. For historical reasons, the h function is called a hypothesis.

Here, we will use $x^{(i)}$ to denote the "input" variables, also called input features, and $y^{(i)}$ to denote the "output" or target variables that we are trying to predict. A pair $(x^{(i)}, y^{(i)})$ is called a training example. We will call a dataset that will be used for training a training dataset -a list of m training examples $\{(x^{(i)}\ , y^{(i)}\ );$ i = 1, . . . , m$\}$.

If we deal with a target variable y that is continuous, we call the learning problem a regression problem. On the contrary, if y can take only discrete values, we call it a classification problem. [27]

In supervised learning our goal is to find a function $y{=}h(x)$ so that we have $y(i){\approx}h(x(i)$ for each training example.

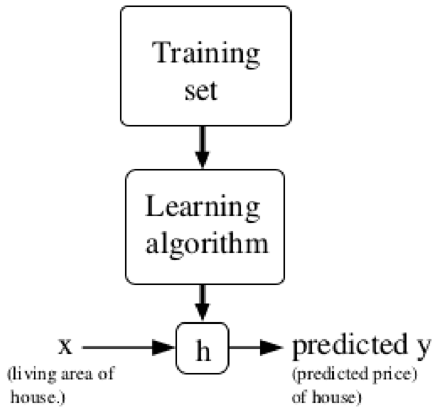


Figure 3.1: Supervised Learning process

3.1.1 Linear regression

A regression problem can often be faced with a linear regression model. In this case we choose a linear function of x:

$$h_\theta(x) = \sum_i (\theta^{(i)} x^{(i)}) = \theta^\top x \tag{3.1}$$

where θ is a vector (θ_0, θ_1, ..., θ_n), and the θ_i 's are the parameters (also called weights) parametrizing the space of linear functions mapping from X to Y.

Here, $h(x)$ represents a large family of functions parametrized by the choice of θ. (We call this space of functions a "hypothesis class".) With this representation for h, our task is to find a choice of θ so that $h(x(i))$ is as close as possible to $y(i)$. In particular, we will choose a θ that minimizes:

$$J(\theta) = \frac{1}{2}\sum_i \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 = \frac{1}{2}\sum_i \left(\theta^\top x^{(i)} - y^{(i)}\right)^2 \tag{3.2}$$

This function, called **mean squared error (MSE)**, is the "cost function" for our problem, and measures how much error is incurred in predicting $y(i)$ for a particular choice of θ. This may also be called a "loss", "penalty" or "objective" function [27]. When J(θ) is minimized the log likelihood l(θ) of p(y |x ; θ), the distribution of $y^{(i)}$ given $x^{(i)}$ parametrized by θ, is actually maximized. This implies that when we chose this particular cost function we relied on the **principle of maximum likelihood** which says that we should choose θ to maximize the likelihood function L(θ).

In order to express p(y|x;θ) as a specific function we need to make some assumptions based on mere intuition. We will firstly assume that the target variables and the inputs are related via the equation:

$$y^{(i)} = \theta^\top x^{(i)} + \epsilon^{(i)}, \tag{3.3}$$

where $\epsilon^{(i)}$ is an error term. Let us further assume that the $\epsilon^{(i)}$ are distributed according to a Gaussian distribution (also called a Normal distribution) with mean zero and some variance σ^2. We have that "e (i) $\sim$ N (0, σ^2)." or "y|x; $\theta \sim$ N (μ, σ^2)". The density of $\epsilon^{(i)}$ is calculated by:

$$P(\epsilon^{(i)}) = \frac{1}{\sqrt{r\pi\sigma}} \exp(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}), \tag{3.4}$$

which can be re-written as:

$$P(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{r\pi\sigma}} \exp(-\frac{(y^{(i)} - \theta^\top x^{(i)})^2}{2\sigma^2}), \tag{3.5}$$

Thus:

$$\begin{aligned} L(\theta) &= \prod_i P(y^{(i)}|x^{(i)};\theta) & (3.6) \\ &= \prod \frac{1}{\sqrt{r\pi\sigma}} \exp(-\frac{(y^{(i)} - \theta^\top x^{(i)})^2}{2\sigma^2}), & (3.7) \end{aligned}$$

and

$$\ell \ (\theta) = \log L(\theta) = n \log \frac{1}{2}\sigma_i\sqrt{r\pi\sigma} - \frac{1}{\sigma^2} \cdot \sum_i (y^{(i)} - \theta^\top x^{(i)})^2, \quad (3.8)$$

Hence, maximizing $\ell \ (\theta)$ (the log likelihood) is equivalent to minimizing

$$\sum_i (y^{(i)} - \theta^\top x^{(i)})^2. \quad (3.9)$$

3.1.2 Logistic regression

Logistic regression is a model used for classification tasks. In contrast to the regression problems the values y that we now want to predict are discrete values. Logistic regression is an algorithm which deals specifically with binary classification problems. More specifically, these are problems in which y can take on only two values, 0 and 1. Given $x^{(i)}$, the corresponding $y^{(i)}$ is called the label for the current training example.

The hypothesis function we choose here is:

$$h_\theta(x) = \frac{1}{(1 + e^{-\theta^\top x})} \quad (3.10)$$

Furthermore:

$$P(y = 1|x;\theta) = h_\theta(x) \quad (3.11)$$

$$P(y = 0|x;\theta) = 1 - h_\theta(x) \quad (3.12)$$

and in a more succinct form:

$$P(y|x;\theta) = (h_\theta(x))^y(1 - h_\theta(x))^{(1-y)} \quad (3.13)$$

Once again following the maximum likelihood principle and by assuming the distribution y|x; $\theta \sim$ Bernoulli(φ) we choose as loss function:

$$J(\theta) = -\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right] \tag{3.14}$$

3.1.3 Softmax Regression

In cases where we are interested in multi-class classification we can employ the softmax regression model. In this case, the output y can get K different values, not only two. Thus, in our training set, we now have that $y^{(i)} \in \{1,2,\ldots,K\}$.

The hypothesis function of softmax regreesion is able to estimate the probability that $P(y{=}k|x)$ for each value of $k{=}1,\ldots,K$, for a given test input. Hence, our hypothesis function will give us as a result a K-dimensional vector, whose elements sum to 1 and each of these values corresponds to one of the K estimated probabilities. Our hypothesis $h_\theta(x)$ can also be represented like this:

$$h_\theta(x) = \begin{bmatrix} P(y = 1|x;\theta) \\ P(y = 2|x;\theta) \\ \vdots \\ P(y = K|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x)} \begin{bmatrix} \exp(\theta^{(1)\top} x) \\ \exp(\theta^{(2)\top} x) \\ \vdots \\ \exp(\theta^{(K)\top} x) \end{bmatrix} \tag{3.15}$$

where $\theta^{(1)},\theta^{(2)},\ldots,\theta^{(K)} \in \Re n$ correspond to our model's parameters.

Similarly, by assuming multinomial distribution, our loss function is:

$$J(\theta) = -\left[\sum_{i=1}^{m}\sum_{k=1}^{K} 1\left\{y^{(i)} = k\right\} \log \frac{\exp(\theta^{(k)\top} x^{(i)})}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x^{(i)})}\right] \tag{3.16}$$

In the equation above, 1{·} is the "'indicator function,"' so that 1{a true statement}=1, and 1{a false statement}= 0.

The above function, commonly called **categorical cross-entropy loss**, generalizes the logistic regression loss function presented in the previous section [27]. We should finally note that in softmax regression, we have that:

$$P(y^{(i)} = k | x^{(i)}; \theta) = \frac{\exp(\theta^{(k)\top} x^{(i)})}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x^{(i)})} \tag{3.17}$$

3.2 Stochastic Gradient Descent

Stochastic Gradient Descent is an algorithm normally used for the training procedure of neural networks. This algorithm takes as input the cost function and the training data set. It calibrates the parameters in such a way that the cost function becomes smaller and smaller until it converges to a value that minimizes the loss. The algorithm works as follows [14]:

```
1: Input: Function f (x; $\theta$) parameterized with parameters $\theta$.
2: Input: Training set of inputs x 1 , . . . , x n and desired outputs y 1 , . . . , y n .
3: Input: Loss function L.
4: while stopping criteria not met do
5:   Sample a training example x i , y i
6:   Compute the loss L(f (x i ; $\theta$), y i )
7:   ĝ $\leftarrow$ gradients of L(f (x i ; $\theta$), y i ) w.r.t $\theta$
8:   $\theta \leftarrow \theta - \eta$ t ĝ
9: return $\theta$
```

This algorithm aims at learning parameters θ so as to minimize the total loss $\Sigma_i L(f(x_i;\theta), y_i)$ over the training set. It relies on sampling epoch by epoch each training example and by computing the gradient of the error on each pair. Next, the parameters θ are updated oppositely from the gradient and is scaled by a learning rate η. The learning rate can either be fixed during the training, or decay as a function of the time step t.

In line 6, SGD calculates the error based on a single training example and it is consequently a vague estimate of the loss that we are trying to diminish. Inaccurate gradients may occur due to the noise in the loss computation. In order to reduce the noise we can employ a sample of training pairs and then estimate the error and the gradient of the batch instead of separated examples. This gives rise to the minibatch SGD algorithm [14]:

```
1: Input: Function f (x; $\theta$) parameterized with parameters $\theta$.
2: Input: Training set of inputs x 1 , . . . , x n and desired outputs y 1 , . . . , y n .
3: Input: Loss function L.
4: while stopping criteria not met do
5:   Sample a minibatch of m examples {(x 1 , y 1 ), . . . , (x m , y m )}
6:   $\hat{g} \leftarrow 0$
7:   for i = 1 to m do
8:       Compute the loss L(f (x i ; $\theta$), y i )
9:       $\hat{g} \leftarrow \hat{g}$ + gradients of 1/m*L(f (x i ; $\theta$), y i ) w.r.t $\theta$
10:   $\theta \leftarrow \theta - \eta$ t $\hat{g}$
11: return $\theta$
```

The gradient ĝ of the loss - based on the minibatch - is calculated in the line 6-9 by repeatedly updating the value of ĝ and finally of θ. The size of the minibatch is m and in general the larger it is the better it can estimate the gradient, but small ones enable faster convergence. It's not however only the accuracy of the gradient calculation that is improved with this new algorithm but also the training efficiency thanks to the ability of parallelizing the calculations, often with the use of GPU[14]. The convergence to global optimum is guaranteed provided that we deal with convex functions, but even in the case of non-convex ones we can get remarkable optimizations without however ensuring convergence to global optimum. A drawback of the minibatch algorithm as compared with the simple SGD is that it can get much slower, since it updates the parameters only after scanning the whole batch.

The gradient computation is a key step for the SGD algorithm, as well as in all other training algorithms. However, we haven't yet dealt with the details about how to compute the gradients of the neural network's error with respect to the parameters. The

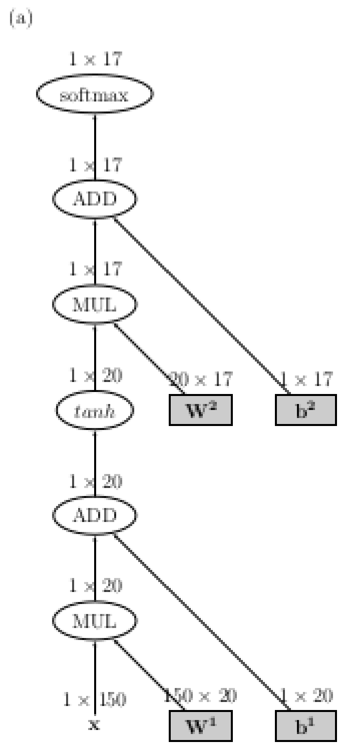


Figure 3.2: Computation graph created for the implementation of backpropagation on a MLP with one hidden-layer and a softmax output transformation [14]

solution comes in the form of an algorithm called backpropagation. Through the use of the chainrule, the backpropagation algorithm takes on the responsibility to methodically compute the required derivatives, while storing intermediary results to the memory [14]. It specifically exploits the computation graph of a training model such as the one shown in the Figure 3.2. We won't, however, get into greater details for this method.

SGD is strongly connected to our work since we ourselves use prediction models such as Word2vec, which are practically neural netwokrs. When training a prediction model, the parametrized function f is the model itself, and the parameters θ are the linear-transformation matrices, bias terms, in our case the **embedding matrices** and so on.

3.2.1 Word Embeddings

If we want to make predictions we need to do computations and a good way to do that is by processing on words as if they were vectors. Instead of vector representations we could also use distinct symbols, but techniques based on this strategy usually suffer for inefficiency and poor generalization. By using vector representations we can represent semantic features and thus infer similarity or other kind of relations between lexical units. The distributional hypothesis that we mentioned above, gives us a good hint on how we can create these representations. Many of the techniques employed till now in the NLP community are based on this very assumption. The first approaches of this kind aimed at encoding the association of words with the contexts in which they appear, either by creating sparse vectors that incorporate this information, or by using clustering techniques.

However, the contemporary approach in tasks that require the use of word representations, is to create vectors with latent features, that is, small and dense vectors that enable fast computations and low memory usage. This strategy is based on machine learning algorithms that involve neural network language models. The creation of such representations depends on training prediction models that can usually be seen as neural networks. Bengio et al. [4] were the first ones who introduced in the NLP community the neural network techniques for the creation of vector representations that they called word embeddings. Collobert and Weston [9] in 2008 showed how useful could such pre-trained embeddings be for linguistic tasks and then it was Mikolov et al. [25] in 2013 that introduced word2vec, a familly of very efficient models for word embeddings, that have shaped, till the current days, an eminent trend in word similarity tasks. The training of all these neural models is based on stochastic gradient descent.

These kinds of models require supervised training, whilst there are some unsupervised (distributional) training techniques that also seem to be quite competitive. SVD, a technique for dimensionality reduction through matrix factorization, has been proposed by Bullinaria and Levy (2007) [8] for linguistic tasks and recently Levy and Goldberg (2014) [24] have argued for a strong connection betweeen word2vec and SVD in terms of their mathematical foundations. In 2014, Pennington et al. [28] released GloVe, a new global log-bilinear regression model that combines the advantages of global matrix factorization and local context window methods.

Yet, the choice of the training set is probably the most decisive factor for the behaviour of our models. As we saw in the corresponding section there are several methods for semantic relation extraction. A pattern-based, as contrasted with distributional approaches in what has to do with the choice of training data, can lead to totally different model behaviours.

There are several software packages for word embeddings creation such as word2vec and Gensim using word-windows based contexts, word2vecf which is a modified version of word2vec allows the use of arbitrary contexts, and GloVe implementing the GloVe model. Many pre-trained word vectors are also available for download on the web.

3.3 Language Models

Language models are algorithms that aim at predicting a word in a phrase, given its n-1 previous words i.e. $p(wt|wt{-}1,\cdots wt{-}n{+}1)$ [29]. The probability of the occurrence (and hence the validity) of a sentence can be estimated through the product of probabilities of each words by applying the chain rule considering the Markov property (the memoryless property of a stochastic process):

$$P(w_1, \cdots, w_T) = \prod_i P(w_i \mid w_{i-1}, \cdots, w_{i-n+1}) \tag{3.18}$$

In neural language models, the probability is estimated through a softmax layer. The objective function in this case is:

$$P(w_t \mid w_{t-1}, \cdots, w_{t-n+1}) = \frac{\exp(h^\top v'_t)}{\sum_{w_i \in V} \exp(h^\top v'_i)} \tag{3.19}$$

The inner product $h^\top v'_i$ represents here the log-probability of the word w_t, which is normalized by the sum of the log-probabilities of all the words in our vocabulary V. The symbol h corresponds to the output vector of the hidden layer in the feed-forward network in the Figure 3.3, while v'_i is the output embedding of the word w, its representation in the weight matrix of the softmax layer. We should note that although v'_i represents the word w_t, it is created separately from the input word embedding v_i.

At this point, we can't but notice the similarity between the above formula and the one presented in the section of softmax regression. Here, our model is parametrized by h.

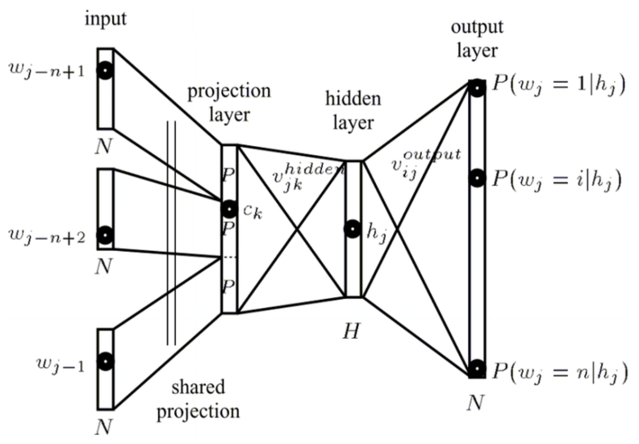


Figure 3.3: The structure of a Neural Language Model

Hence, we could write that:

$$P(w_t \mid w_{t-1}, \cdots, w_{t-n+1}) = P(y^{(i)} = k|x^{(i)}; h) \tag{3.20}$$

The cost function can be derived as seen below:

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} \log P(w_t \mid w_{t-1}, \cdots, w_{t-n+1}) \tag{3.21}$$

We can see that we need to calculate the probability of every word w at the output layer of the neural network. To do this efficiently, we perform a matrix multiplication between h and a weight matrix whose rows consist of v_i' of all words w in V. We then give the resulting vector, i.e. the output of a previous layer, as input to the softmax layer and this, in turn, transforms the vector to a probability distribution over the words in V.

The reason that we describe the functioning and the structure of neural language models, in which a network is trained to predict the next word based on a sequence of preceding words, is because Neural word embeddings originated from the last one. The big step for the emergence of the concept of word embedding is to stop caring about predicting features of language models, to concentrate in the resulting parameters and finally to ignore the constraint of caring merely for the previous words of the target word and perceive the context as a symmetric window around the focus word [29].

3.4 Word2vec

It was imperative to explain the softmax and logistic regression model and its use in language models in order to have a deep understanding of Mikolov's word2vec architectures [25]. Word2vec, as mentined before, is a family of models that are used to produce word embeddings. These models represented are shallow, two-layer neural networks that are trained to predict linguistic contexts given a target word (Skip-gram) and inversely (CBOW). An abstraction of each model's functioning can be seen in Figure 3.4.

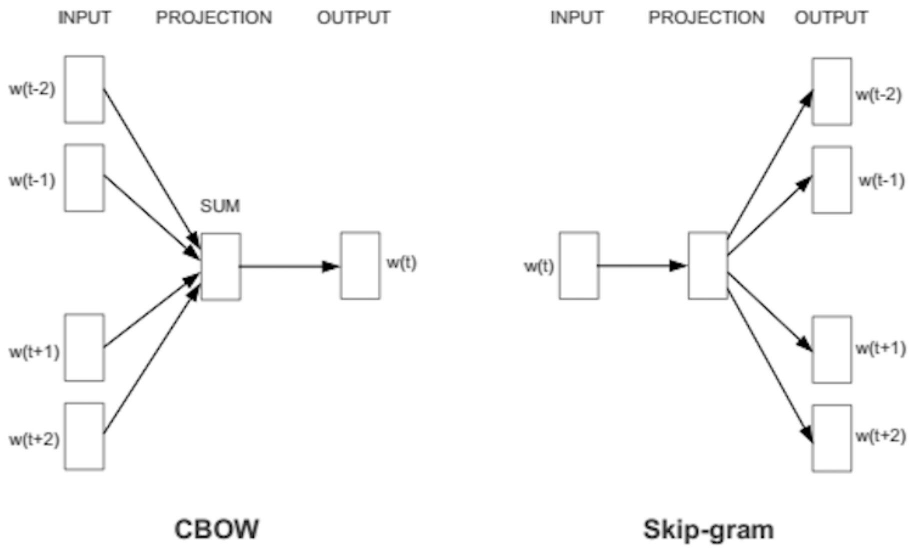


Figure 3.4: CBOW and Skip-gram abstract picturing [4]

Word2vec usually takes as input a large corpus, a set of discrete sentences, and produces vector representations of words, typically of several hundred dimensions. According to the authors' note, CBOW is faster while skip-gram is slower but does a better job for infrequent words.

An example of a sentence would be: "The quick brown fox jumps over the lazy dog." In Figure 3.5 we can see different training samples with the target word highlighted with blue colour and the context words framed by a window of size C=2.

In general, feed-forward neural networks for NLP tasks are fed with words in the input layer (usually represented in the form of one-hot vectors) and nextly these words are embedded as dense vectors. These vectors, which are learnt through back-propagation, constitute the models' parameters. In the case of Word2vec our vocabulary is embedded both in the weights of the input layer (i.e. input vectors) and in the weights of the output layer (i.e. output vectors).

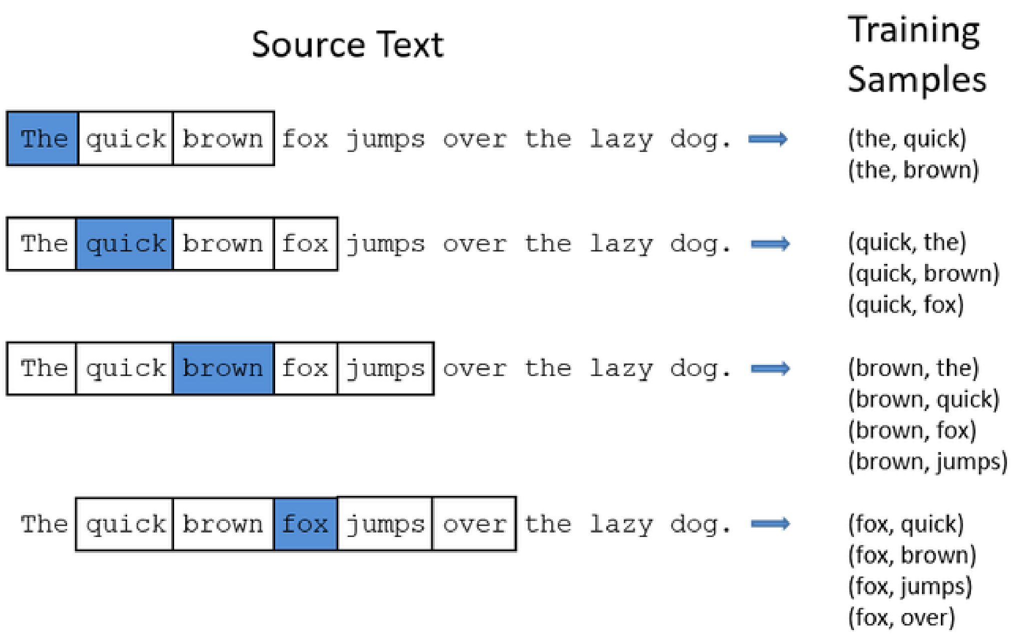


Figure 3.5: Example of of Word2vec's processing on a sentence

There is, however, a big difference between Neural Networks such as the neural language models shown in Figure 3.3, where the vector space emerges more or less as a byproduct of the training procedure, and models such as Word2vec where learning these vectors constitute its ultimate goal from the very beginning. In contrast to Word2Vec, regular neural networks usually produce task-specific embeddings with very limited use in general tasks. Another very important advantage of Word2Vec as compared with other Neural Networks is its ability to minimize the computational complexity of the training phase due to its simple structure, i.e. the lack of non-linearities since it contains no hidden layer. It is, thus, wrong to consider Word2vec to be part of deep learning, as its architecture is neither deep nor uses non-linearities.

Here, we will express the training complexity of Word2vec family of models as:

$$O = E \times T \times Q, \tag{3.22}$$

where E is the number of the training epochs, T is the number of the words in the training set (the number of our samples) and Q is a quantity that will be defined separately for each model architecture of Word2vec. Q is what we actually try to minimize through model's structure modifications.

3.4.1 Continuous Bag-of-Words Model

This architecture consists of input, projection and output layers as shown in the Figure 3.6.

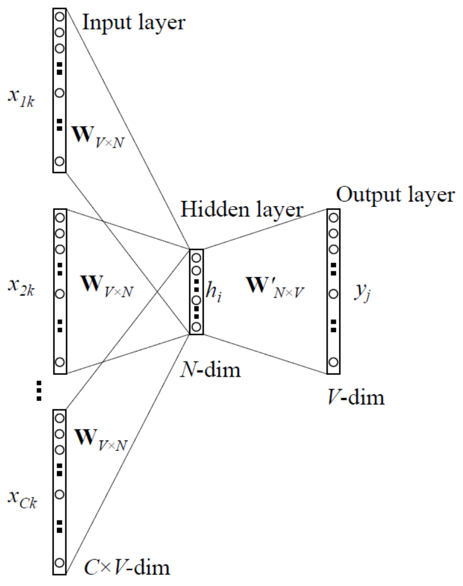


Figure 3.6: CBOW's architecture

Before we start training our model, we have to find a way to represent our words in such a form that we can feed them into the input layer of our Neural Network. We choose to represent them as on-hot vectors which are sparse vectors of dimensionality V (our vocabulary size). For example, the one-hot vector of the third word of our vocabulary would be:

$$w_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.23}$$

With C we will refer to our window size. The input one-hot vectors can be represented as x_i and the output as y_j or simply as y since in this case we deal only with one word in the output layer. The unknown parameters of out model are represented in our two weight matrices, the input matrix $W_{in} \in R^{n\times|V|}$ and the output matrix $W_{out} \in R^{|V|\times n}$, where n is an arbitrary size which defines the size of our embedding space, i.e. the dimensionality of our word embeddings. The i-th column of W_{in} is the n-dimensional vector for the input word w_i. We denote this n × 1 vector as v_i. Similarly, the j-th row of W_{out} is an n-dimensional output vector for w_j. We denote this row of W_{out} as v_i. During the training phase we learn two vectors for every word w_t, i.e. the input word embedded vector v_i and output word embedded vector v'_i.

In Neural Network terms, our input layer is projected to our projection layer that has dimensionality 2C × n, where n denotes the dimensionality of our embeddings. This is how we get our embedded word vectors for a given context. In mathematical terms we have:

$$v_{t-C} = W_{in} \times (t - C), \tag{3.24}$$
$$v_{t-C+1} = W_{in} \times (t - C + 1), \tag{3.25}$$
$$\dots, \tag{3.26}$$
$$v_{t+C} = W_{in} \times (t + C). \tag{3.27}$$

All words get projected into the same position; their vectors are averaged like this:

$$H = \frac{v_{t-C} + v_{t-C+1} + \dots + v_{t+C}}{2C}. \tag{3.28}$$

The calculations done for the composition of the projection layer are not costly at all, since only 2C inputs are active at each epoch. This architecture is called a bag-of-words model because the projection is independent of the order of words in the history.

Finally, in the output layer, we get a transformation of this kind:

$$\hat{y} = softmax(W_{out} \cdot H), \tag{3.29}$$

apparently by employing the softmax method, as described above, or approximations of it that we are going to examine afterwards.

We aim at minimizing the error between the estimated probabilities, ŷ and the true probabilities, y, where y is the one-hot vector of the actual word. Here, $y = y_1, \ldots, y_n$ denotes a vector representing the true multinomial distribution over the labels 1, ... , n, and $\hat{y} = \hat{y}_1, \ldots, \hat{y}_n$ the network's output, which was transformed by the softmax activation function, and represent the conditional distribution $\hat{y}_i$ = P (y = i|h).

The **categorical cross entropy loss** is a good measure of the dissimilarity between the true distribution y and the predicted distribution ŷ. It is the same loss function presented in the section about softmax, here expressed in this simple form:

$$\sum_i y_i \log \hat{y}_i = softmax(W_{out} \cdot H), \tag{3.30}$$

and since we use one-hot vectors, the loss is simplified to:

$$y_i \log \hat{y}_i = softmax(W_{out} \cdot H), \tag{3.31}$$

At this point, i refers to the index where the correct word's one hot vector is 1. Furthermore, we notice that our cost function is slightly different from the one of language model and can be formulated like this:

$$\begin{align}
J(\theta) &= -\text{log}P(w_t \mid w_{t-C}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+C}) \tag{3.32}\\
&= -\text{log}P(v'_t \mid H) \tag{3.33}\\
&= -\text{log}\frac{\exp(v'^{\top}_t H)}{\sum_{i=1}^{V}\exp(v'^{\top}_t v_i)} \tag{3.34}\\
&= -v'^{\top}_t H + \text{log}\sum_{i=1}^{V}\exp(v'^{\top}_t v_i) \tag{3.35}
\end{align}$$

This model is named after CBOW, as unlike standard bag-of-words model, it uses continuous distributed representation of the context.

Returning to the discussion about the complexity issue, by employing the softmax method the term Q becomes:

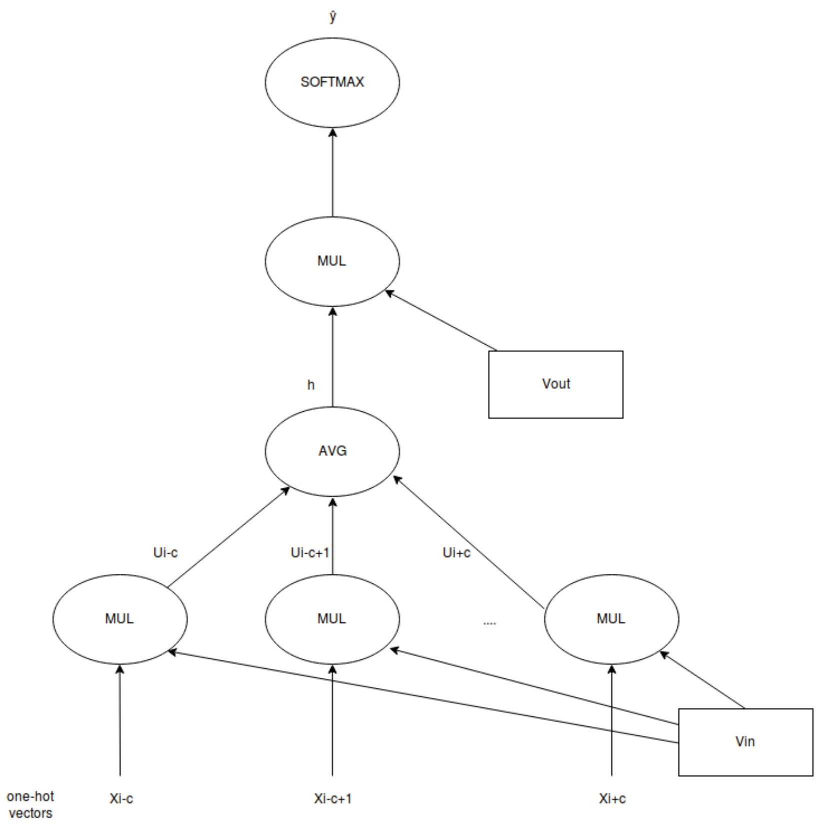


Figure 3.7: CBOW computation graph

$$Q = C \times n + n \times V \tag{3.36}$$

The first term refers to the normalization taking place in the projection layer and the second refers to the softmax function applied in the output layer. Finally, using approximations of softmax such as hierarchical softmax, the training complexity is reduced to:

$$Q = C \times n + n \times log_2 V. \tag{3.37}$$

Our model is trained using gradient descent and back-propagation in order to change both the input and the output vectors.

3.4.2 Continuous Skip-gram Model

This architecture is quite similar to CBOW, but instead of aiming at the prediction of a word by its context, it learns to predict the context given the target word. The representations created in Skip-gram emerge by repeatedly feeding our linear (or log-linear as we will see soon) classifier with one-hot input vectors which are then projected to the next layer similarly to the CBOW without however the need of the calculation of the average, since we only deal with one input vector per epoch. We then try to predict the surrounding words within a range determined by the window size we choose [25].

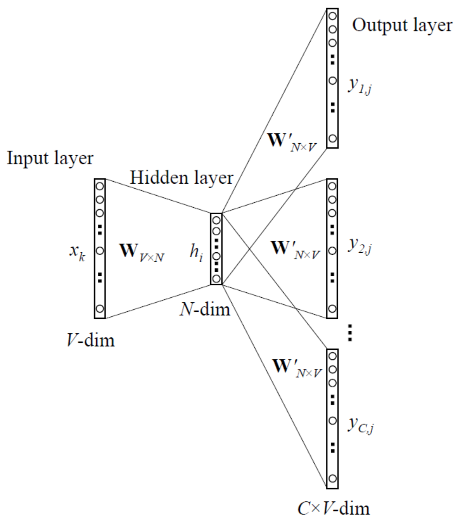


Figure 3.8: Skip-gram's architecture

The input one-hot vector will be represented similarly to the CBOW's case with an x, yet without any index since we need only one input word at a time. The output vectors are y_j. We define W_{in} and W_{out} the same as in CBOW. Similarly as in the case of CBOW, we generate our one-hot input vector x and we get our word embedding:

$$H = u_t = W_{in}x. \tag{3.38}$$

Once again we employ a softmax layer:

$$y = softmax(W_{out} \cdot H). \tag{3.39}$$

We desire our predicted probability distribution y to match the true vector y which is produced as the sum of the 2C one-hot vectors surrounding words and hence contains only 2C ones and V-2C zeros:

$$y = y_{t-C} + \cdots + y_{t-1} + y_{t+1} + \cdots + y_{t+C}. \tag{3.40}$$

As in CBOW, we need to employ a cost function in order to evaluate the model. Here we invoke a Naive Bayes, i.e. a strong (naive) conditional independence assumption. Said differently, we assume that given the target word, all context words are completely independent:

$$\begin{aligned}
J(\theta) &= -\log P(w_{t-C}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+C} \mid w_t) & (3.41)\\
&= -\log \prod_{-C \le j \le C, \ne 0} P(w_{t+j} \mid w_t) & (3.42)\\
&= - \sum_{-C \le j \le C, \ne 0} \log P(w_{t+j} \mid w_t) & (3.43)\\
&= - \sum_{-C \le j \le C, \ne 0} \log P(v'_{t+j} \mid v_t) & (3.44)\\
&= - \sum_{-C \le j \le C, \ne 0} \log \frac{\exp({v'_{t+j}}^\top v_t)}{\sum_{i=1}^{V} \exp({v'_i}^\top v_t)} & (3.45)\\
&= - \sum_{-C \le j \le C, \ne 0} {v'_{t+j}}^\top v_t + 2C \log \sum_{i=1}^{V} \exp({v'_i}^\top v_t) & (3.46)
\end{aligned}$$

The training complexity of this architecture is:

$$Q = C \times (n + n \times V). \tag{3.47}$$

With the assumption that we use an approximation of softmax like the ones we will see in the sections to come, the complexity becomes:

$$Q = C \times (n + n \times log_2 V). \tag{3.48}$$

3.4.3 Digression: From Brain-Inspired representations to mathematical abstraction

In the previous sections, while studying CBOW and Skip-gram model we were mostly thinking in terms of layers. This approach of Neural Networks derives from the convenience of graphical representation of our training models. This way we have a good insight of each separate component of the algorithms and at the same time we stick to the Brain-Inspired graphical representation of ANNs, the basic source of inspiration of models such as Multilayer Perceptron (MLP). In this type of representation of MLP, a neuron constitutes a computational unit that has scalar inputs and outputs. Each input is associated with a parameter called weight. The neuron multiplies each input by its weight, it sums them and then applies to it a non-linear (sigmoid) function to produce the output [14]. The neurons are interconnected, forming a network of the following form:

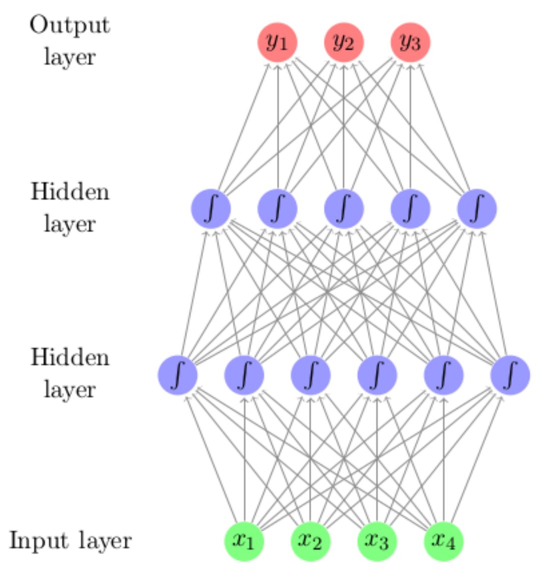


Figure 3.9: Multilayer Perceptron with two hidden layers (MLP2) [14]

Given that a neural network has enough neurons and a non-linear activation function, it can approximate a huge amount of mathematical functions. It has been proven for example that MLP1 - hence, every feed-forward network with linear output layer and at least one hidden layer with a “squashing” activation function - is a universal approximator, i.e. it can can approximate any Borel measurable function from one finite dimensional space to another [14].

Function is the keyword here, since this is exactly what a Neural Network is and it can be represented as one, i.e. as a hypothesis function. For example, the MPL1 can be written as [14]:

$$NN_{MLP1}(x) = g^1(xW^1 + b^1)W^2 + b^2 \tag{3.49}$$

and the MLP2 shown in Figure 3.9 as [14]:

$$NN_{MLP1}(x) = g(xW^1 + b^1)W^2 + b^2 \tag{3.50}$$
$$h^1 = g^1(xW^1 + b^1) \tag{3.51}$$
$$h^2 = g^2(xW^2 + b^2) \tag{3.52}$$
$$y = h^2W^3 \tag{3.53}$$

This representation is very powerful and succinct. It is very rarely in the literature that we can find a mathematical abstraction of Word2vec models. Yet, considering the rigorous presentation made above we are but just one step before we do this. Here it is:

$$h_{w2v}(word_x) = h'_{w2v}(x_{one-hot}) = h^{w_{out}}_{softmax}(W \cdot x) = h^{w_{out}}_{softmax}(v_x), \tag{3.54}$$

$$h'_{w2v}(x) = \begin{bmatrix} P(word_{y_1}|v_x; W_{out}) \\ P(word_{y_2}|v_x; W_{out}) \\ \vdots \\ P(word_{y_V}|v_x; W_{out}) \end{bmatrix} = \frac{1}{\sum_{j=1}^{V} \exp(W^{(j)\top}_{out} v_x)} \begin{bmatrix} \exp(W^{(1)\top}_{out} v_x) \\ \exp(W^{(2)\top}_{out} v_x) \\ \vdots \\ \exp(W^{(V)\top}_{out} v_x) \end{bmatrix} \tag{3.55}$$

The difference between CBOW and Skip-gram is only the choice of v_x, since in the first case it constitutes the average of all the input context words and in the second case it is merely the embedding of the input target word.

3.4.4 Softmax Approximation Strategies

In this section we will discuss different strategies that have been proposed to approximate the softmax function. These approaches can be grouped into softmax-based and sampling-based approaches [30]. Softmax-based approaches are methods that keep the softmax layer intact but modify its architecture to improve its efficiency. Here, among all the softmax-based approximation algorithms we will examine only the **Hierarchical Softmax**. Sampling-based approaches on the other hand completely do away with the softmax layer and instead optimise some other loss function that approximates the softmax. We will focus on one algorithm of this kind called **Negative Sampling**.

3.4.4.1 Softmax-based Approaches: Hierarchical Softmax

Hierarchical softmax is considered to be a computationally efficient approximation of softmax, since it reduces the size of the output layer from V to $log_2 V$. In common NLP tasks the use of hierarchical softmax can accelerate the training phase at least 50× [26].

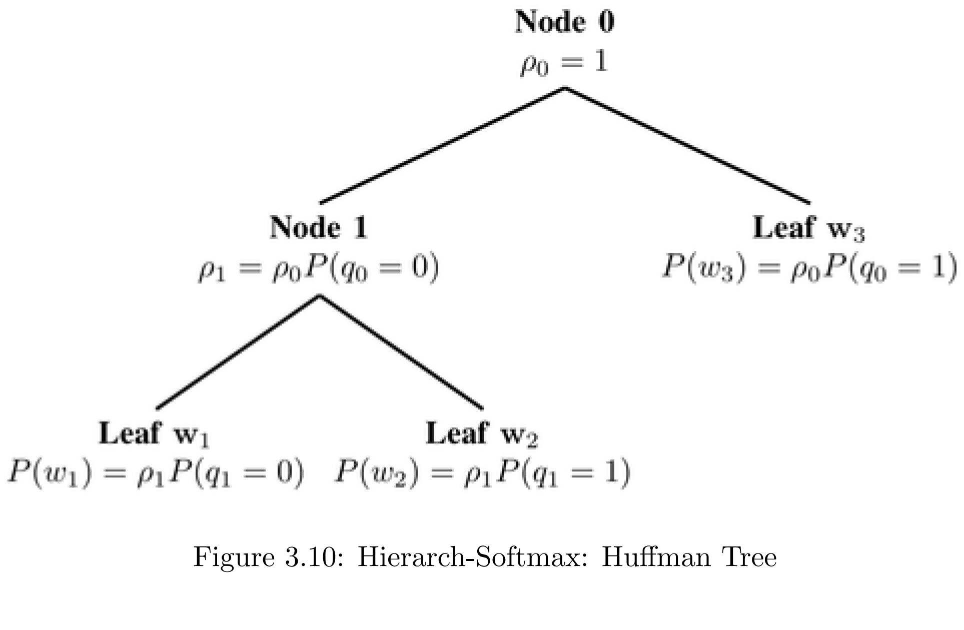

Node 0
$\rho_0 = 1$
Node 1
$\rho_1 = \rho_0 P(q_0 = 0)$
Leaf w_3
$P(w_3) = \rho_0 P(q_0 = 1)$
Leaf w_1
$P(w_1) = \rho_1 P(q_1 = 0)$
Leaf w_2
$P(w_2) = \rho_1 P(q_1 = 1)$

Figure 3.10: Hierarch-Softmax: Huffman Tree

The hierarchical softmax uses a binary tree representation of the output layer. Its leaves correspond to the V words of our vocabulary and, in each node, we store the relative probabilities of its child nodes. Thus, each leave can be reached through a path for the root of our tree structure. This caching technique allows us to decompose the calculation of the final probability of one word into a sequence of probability calculations. Thus, we speed up the procedure because it's not any more necessary to calculate the expensive normalization over all words. More specifically, Mikolov et al. used a binary Huffman tree as the one shown in Figure 3.10, because "it assigns short codes to the frequent words which results in fast training" [26].

The regular softmax can be thought of as a tree of depth 1, with each word in V represented as leaf. The problem is that we have to do V times (for all our word-leaves) a normalization through this not negligible calulation:

$$P(word_{y_j}|v_i; W_{out}) = \frac{\exp(W_{out}^{(j)\top} v_i)}{\sum_{k=1}^{V} \exp(W_{out}^{(j)\top} v_i)} \tag{3.56}$$

By employing hierarchical softmax we reduce the calculations to the $log_2 V$ which is the height of our binary tree.

One big difference between hierarchical softmax and "vanilla" Skip-gram is that in the first case we only need one matrix for our vector representations in contrast to the second case where we need both an input and an output matrix. In our project, though, we strongly rely on both input and output representations of each word. That is what makes H-softmax method inadequate for our task and thus we will restrict ourselves from further analysing its functioning.

3.4.4.1.1 Sampling-based Approaches: Negative Sampling The Sampling-based approaches get rid of the softmax layer by introducing a much cheaper method instead of the costly normalization technique used in softmax. However, it's only during the training phase that we can use these methods. That is not a problem as long as we don't use our model explicitly for predictions and we only care about the learned parameters, i.e. our word embeddings.

The first Sample-based model we are going to look at is Noise Contrastive Estimation (NCE). NCE exploits logistic regression for biinary calssification. For every word w it generates k noise samples - something like a fake context - from a noise distribution Q. The probability that a word together with its context (w,c) came form the initial corpus is $P(y = 1|w, c)$ and the probability that it didn't is $P(y = 0|w, c)$.

We represent the probability of sampling a positive or a noise sample as following:

$$P(y, w \mid c) = \frac{1}{k+1} P_{\text{train}}(w \mid c) + \frac{k}{k+1} Q(w) \tag{3.57}$$

The probability of predicting a positive example is:

$$P(y = 1 \mid w, c) = \frac{P(w \mid c)}{P(w \mid c) + k\,Q(w)} = \frac{\exp(h^\top v'_w)}{\exp(h^\top v'_w) + k\,Q(w)}, \tag{3.58}$$

while predicting a negative one is naturally its complement:

$$P(y = 0 \mid w, c) = 1 - P(y = 1 \mid w, c). \tag{3.59}$$

Now we pass to the next sampling-based model called Negative Sampling (NEG), which is actually the one used by word2vec as an alternative to hierarchical softmax. NEG further approximates the probability distribution produced by NCE by making it as fast to compute as possible. For this reason, it sets the most expensive term, $k\,Q(w)$ to and so:

$$P(y = 1 \mid w, c) = \frac{\exp(h^\top v'_w)}{\exp(h^\top v'_w) + 1} = \frac{1}{1 + \exp(-h^\top v'_w)} \tag{3.60}$$

Since Skip-gram model is merely concerned with learning vector representations, we can simplify NCE as long as the quality of the vector representations is not distorted. By deciding to use logistic regression just like in NCE, our goal is to minimize our loss function, i.e. the negative log-likelihood (or cross-entropy) of our training examples against the noise.

$$J_\theta = -\sum_{w_i \in V} [\log \frac{1}{1 + \exp(-h^\top v'_{w_i})} + \sum_{j=1}^{k} \log (1 - \frac{1}{1 + \exp(-h^\top v'_{\tilde{w}_{ij}})}] \tag{3.61}$$

$$= -\sum_{w_i \in V} [\log \frac{1}{1 + \exp(-h^\top v'_{w_i})} + \sum_{j=1}^{k} \log (\frac{1}{1 + \exp(h^\top v'_{\tilde{w}_{ij}})}] \tag{3.62}$$

And by setting $\sigma(x)$=1/(1+exp($-x$)) we get the NEG loss function as presented by Mikolov et al. [26]

$$J_\theta = -\sum_{w_i \in V} [\log \sigma(h^\top v'_{w_i}) + \sum_{j=1}^{k} \log \sigma(-h^\top v'_{\tilde{w}_{ij}})] \tag{3.63}$$

The parameters θ - in this case our input and output matrix - for which the loss function is minimized can be represented straightly as:

$$\operatorname*{argmax}_\theta \sum_{w_i \in V} [\log \frac{1}{1+\exp(-h^\top v'_{w_i})} + \sum_{j=1}^{k} \log{(1 - \frac{1}{1+\exp(-h^\top v'_{\tilde{w}_{ij}})}}] \tag{3.64}$$

3.4.5 Subsampling

Word2vec gives the option to get rid of words that occur in high frequencies by randomly removing those that appear more often than a threshold f with a probability p=1$-\sqrt{}$1/f [26].

3.5 Mutual Information Measures

Mutual information is defined as:

$$I(X;Y) = \sum_{x,y} P_{XY}(x,y) \log \frac{P_{XY}(x,y)}{P_X(x)P_Y(y)} = E_{P_{XY}} \log \frac{P_{XY}}{P_X P_Y}. \tag{3.65}$$

Mutual information measures have been studied and used extensively in research topics pertinent to the process of discovering typical lexical associations between words. For a pair of co-occurring items x and y, Pointwise Mutual Information (PMI) is defined as the logarithmic ratio of their joint probability to the expected joint probability if x and y were independent:

$$PMI(X;Y) = \log \frac{P_{XY}(x,y)}{P_X(x)P_Y(y)} = E_{P_{XY}} \log \frac{P_{XY}}{P_X P_Y}. \tag{3.66}$$

PMI is in practice often replaced with positive PMI (PPMI) which replaces negative values with 0 and is defined as:

$$PPMI(X;Y) = max(PMI(X;Y),\ 0) \tag{3.67}$$

One weakness of PMI is that it is prone to overestimating low-frequency data. A variant introduced by Bouma [6] normalizes PMI for smoother results:

$$NPMI(X;Y) = \frac{PMI(X;Y)}{-log_2 P(X;Y)} \tag{3.68}$$

3.6 Unsupervised Learning

In unsupervised learning we give an unlabeled training set to an algorithm and we ask the algorithm to find some structure in the data for us. More accurately, it is the machine learning task of inferring a function to describe hidden structure from unlabeled data . Since the examples given to the learner are unlabeled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm. This is one way of distinguishing unsupervised learning from supervised and reinforcement learning.

The most common unsupervised learning models are related with clustering. During the training phase, they create different clusters for the inputs and henceforth any new input can be categorized in its appropriate cluster. Other than clustering, some unsupervised learning techniques are: anomaly detection, Hebbian Learning and learning latent variable models such as Expectation-Maximization algorithm, Method of moments (mean, covariance) and Dimentionality Reduction (Principal component analysis, Independent component analysis, Non-negative matrix factorization, Singular value decomposition).

3.6.1 Dimentionality Reduction

The idea behind dimensinality reduction is that, sets of data that are represented by matrices of size $n \times m$ can be summarized by smaller matrices. Operations on these small matrices can approximate the initial one. Naturally, the new matrices have either less rows or less columns compared with the ones they approximate and hence can be manipulated much more efficiently.

3.6.1.1 Principal Component Analysis (PCA)

Principal Component Analysis, or PCA, is a popular technique used for applications such as dimensionality reduction, data compression, feature extraction, and visualization [20]. PCA is defined as the orthogonal projection of the data onto a lower dimensional linear space, known as the principal subspace, such that the variance of the projected data is maximized [19]. PCA relies strongly on Gaussian features. Singular Value Decomposition (SVD) is often employed as an efficient method to calculate the desired principal components.

3.6.1.2 Independent Component Analysis (ICA)

Independent Component Analysis (ICA) is a variant of PCA that retains linearity but does away with the requirement for Gaussian distribution. There are several cases where real-world have non-Gaussian features. A common use is for the solution of the blind sourve separation problem. [5]

3.6.1.3 Matrix Factorization

The objective of matrix factorization is to take an input matrix A and find an (approximately) equivalent representation of it by using the product of other (smaller) matrices. Among several matrix decomposition techniques, here, we will study only SVD.

3.6.1.4 Singular Value Decompostion(SVD)

SVD is one of the most popular methods for matrix factorization and found its place into NLP via latent semantic analysis (LSA). This method decomposes a matrix A into three matrices $U_1 \Sigma V_1$. Generally, only the top d elements of the new matrices are used when we deal with problems that require dimensionality reduction.

Assuming a rectangular m × n matrix A with real entries and with rank equal to r, there exists an m × m real orthogonal matrix U and an n × n real orthogonal matrix V such that:

$$A = UDV',\ where\ D = \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix},\ and\, \Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r \end{bmatrix} \tag{3.69}$$

where D is m × n, Σ is r × r and the σ i 's are real numbers such that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$. This decomposition is also expressed using the following partition:

$$A = [U_1 : U_2] \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1' \\ V_2' \end{bmatrix} = U_1 \Sigma V_1' \tag{3.70}$$

where U_1 and V_1 are m × r and n × r matrices, respectively, with orthonormal columns and the O submatrices have compatible dimensions for the above partition to be sensible.
The diagonal entries σ i of Σ are known as the **singular values** of A. The columns of U and the columns of V are called the **left-singular vectors** and **right-singular vectors** of A, respectively.

SVD behaves similarly to spectral decomposition in the sense that both methods give us orthonormal bases with respect to which the transformation of A can be represented by a diagonal matrix. In the case of spectral decomposition, A is a real-symmetric matrix, whereas in the case of SVD, A is rectangular [2].

Our orthogonal matrices $U = [u_1 : u_2 : ... : u_m]\, and\, V = [v_1 : v_2 : ... : v_n]$ are chosen so that:

$$AV = [Av_1 : Av_1 : \cdots : Av_r : Av_{r+1} : \cdots : Av_n] \tag{3.71}$$

$$= [\sigma_1 u_1 : \sigma_2 u_2 : \cdots : \sigma_r u_r : 0 : \cdots : 0] = [U_1 : U_2] \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \tag{3.72}$$

Numerical approach

For the computation of SVD there are typically two procedures employed. This takes $O(mn^2)$ floating-point operations, where m $\geq$ n. In the second step we compute the SVD of the bidiagonal matrix with an iterative method. Normally, it suffices to compute the SVD up to a certain precision.

The first step can be done using Householder reflections [32]. For the second step we can use a variant of the QR algorithm for the computation of eigenvalues, as described by Golub and Kahan [15].

3.6.1.5 Truncated SVD

In truncated SVD, which is the method we used in our project (implemented in sci-kit learn), only the first d columns of U_1 and the first d rows of V_1' (that correspond to the largest singular values if Σ) need to be calculated. By discarding the rest of the matrices we employ a much faster model than the classic SVD assuming that $d << r$. The cost that we pay is that we do not achieve an exact factorization of A, but this does not constitute a problem in cases where our main concern is the high generalization of our model. The matrix U_d is thus $m \times d$, Σ_d is $d \times d$ diagonal, and V_d' is $d \times n$.

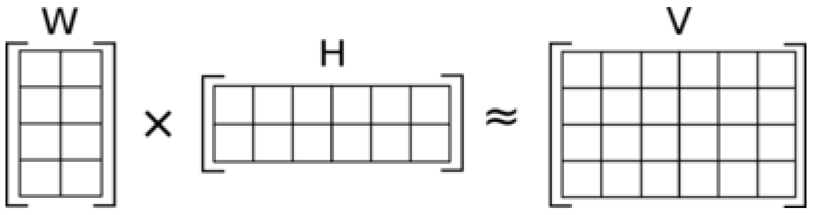
3.6.1.6 Non-negative matrix factorization

With the name Non-negative matrix factorization (NMF or NNMF), we refer to a group of algorithms in multivariate analysis and linear algebra where a matrix A is factorized into two matrices W and H, with all of the matrices' elements being non negative elements. This non-negativity makes the resulting matrices easier to inspect. Since the problem is NP-Hard, it is commonly approximated numerically.

Let matrix **A** be the product of the matrices **W** and **H**,

$$V = WH \tag{3.73}$$

The dimensions of W H must be significantly lower than those of A. Hence, NMF does dimensionality reduction by generating factors with significantly reduced dimensions compared to the original matrix.



We approximate A by minimizing the cost function:

$$\min_{W,H} |A - WH|, where\, W \geqslant 0, H \geqslant 0 \tag{3.74}$$

Chapter 4

Creation of Causality Detection Models

In this chapter we are going to describe how we built our models for the causality detection task. At first, we will describe the process of training data extraction and then we will present the training process and the special architectures that we employed.

4.1 Extraction of Training Data

As a resource for the creation of our training data we used the french annotated frWac corpus, from the WaCky set of corpora [3]. FrWac contains 1.6 billion words and was collected on the Web on the .fr domain. It is thus indeed very large and covers very diverse domains. The corpus includes texts extracted from blogs, which implies that some parts contain many spelling and grammatical errors. As a result, the annotations include a considerable amount of noise that we have to deal with [10].

Our method of extracting words bearing causal meaning relies on the syntactic dependencies of sentences, the part-of-speech tags and the lemmas of words. The syntactic structure of each sentence is represented by a dependency tree in the CoNLL format. More specifically in our case, the syntactic relations are represented through dependency-based parse trees. In order to obtain these, three operations need to be performed: part-of-speech tagging, lemmatization and dependency parsing. Dependency structures consist of lexical items, linked by binary asymmetric relations called dependencies. The dependency trees for the frWaC corpus were obtained using the Bonsai tool, which includes a part-of-speech tagger and lemmatizer, Melt and the MaltParser , trained on the French Treebank, for syntactic parsing. The resulting dependency trees are in the CoNLL format.

Annotations are encoded in plain text files (UTF-8, using only the LF character as line break) with three types of lines:

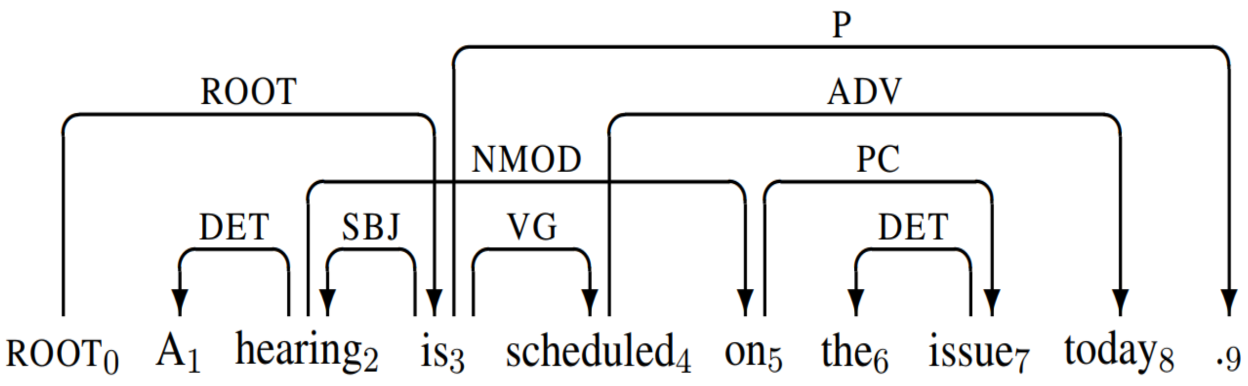


Figure 4.1: Example of n english sentence's Depandency Tree

- Word lines containing the annotation of a word/token in 10 fields separated by sing tab characters.
- Blank lines marking sentence boundaries.
- Comment lines starting with hash (#).

Sentences consist of one or more word lines, and word lines contain the fields: ID, FORM, LEMMA, UPOSTAG, XPOSTAG, FEATS, HEAD, DERPEL, DEPS

The most important units of the extraction process is a set of indicators of causality. One can notice that, at this exact point, we rely on the pattern-based paradigm of relation extraction aiming at creating a robust training set for our machine learning algorithms. The causality indicators are lexical units retrieved from the ASFALDA French FrameNet. FrameNet project provides a structured set of prototypical situations, called frames, along with a semantic characterization of the participants of these situations called roles. For our extraction procedure, we employed the following frames from the Causality domain: Causation, Evidence, Explaining_the_facts, FR_Attributing_cause, FR_Cause_enunciation, FR_Cause_to_start-Launch_process, FR_Contingency-Objective_influence, FR_Reason, Make_possible_to_do, Preventing, Response. We also used the frame FR_Means_for_purpose as far as its corresponding lexical units appear in very similar contexts with the other causality frames although it doesn't strictly belong to the causality domain. However, to simplify matters we call all of the above indicators ***causality triggers***. The lexical units that we kept are those that we didn't consider highly ambiguous.

We separated the triggers into two different categories. The ones that appear in sentences in the form **cause-indicator-effect** and the ones that appear in the form **effect-indicator-cause**. We worked on the assumption that the occurrence of a non-ambiguous indicator in a sentence indicates the existence of a causality relation. Hence, we chose to process only on sentences that contain at least one such lexical unit.

We managed to scan our corpus for such triggers quite efficiently. We unavoidably, of course, processed on each and every line of the frWac corpus (each line gives us information about a specific word). Then, we checked if the lemma belongs in our trigger set by employing binary search in our 64 line trigger list. This way we eliminated the scanning time from Nx64 to Nxlog64=Nx6, where N(=1.6 billion) is the number of the frWac lines in total.

For each occurrence of a causality trigger in a sentence, we then had to retrieve the useful information: the two components of the causality relation (**causal components**), *tuples of meaningful sets of words that represent either the cause or the effect of the captured frame*. For this purpose we rallied three different seed patterns based on the part-of-speech of the employed triggers: **prepositions**, **conjunctions** and **verbs**.

An important constraint about a processed sentence to be considered as a valid source of training data information was for the trigger not to be the (syntactic) root. Ensuring that, by taking advantage of the (syntactic) dependency tree we can find the parent and child nodes of the current trigger. The head is indicated in the corresponding CONLL field, but, in order to find the children we should scan all the words appearing in our sentence and take under consideration those that have as head number the ID of our parent-word. However, we will see afterwards that we needed to have access on children of many other words appearing in our sentence except form the trigger-word, so that we process deeper on the syntactic tree and thus create larger training sets. Considering the above need, we decided to represent each sentence, while processing on it, as an adjacency list of the words' IDs. This way we reassured an efficient and easy access to every word's syntactic child. We later used this structure to implement Best First Search algorithm for the extraction of our causal components.

Here we present the basic patterns used for the extraction

1. Conjunction triggers

In this case, the most common and less ambiguous form that causality relation occurs in a sentence is:

Verb Phrase (effect) - Conjunction - Verb Phrase (cause)

We consider our extraction valid if the parent node is a verb and the child node is a noun. Then we have to analyze the verb phrase in the form Subject - Verb - Object. A problem that arises is what we should consider as constituent words of our Subject and Object. By ignoring stopwords, for each event, we can explore the Subject and Object subtrees using a BFS algorithm since each set of extracted components, what we shall call **causal triplets**, can be structured as a (causal) dependency tree (a subtree of the whole sentence dependency tree) derived from syntactic relations.

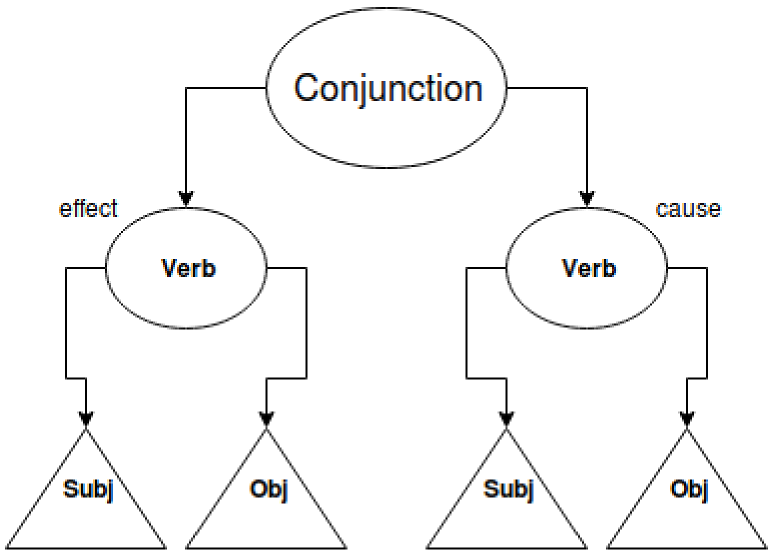


Figure 4.2: Pattern for conjunction triggers

If we have multiple valid children (nouns) of our conjunction trigger, in this case, more than one verb-child, we have to deal with another instance of causal relation in our sentence. We choose to extract more than one triplets from the same sentence, one for every valid child. We chose to store all of the above extracted information to an XML file for simpler and faster further manipulation and visualization.

2. Preposition triggers

This time the causality relation emerges as:

Verb Phrase (effect) – Preposition – Noun Phrase (cause)

The Noun Phrase is regarded as a subtree with the trigger's child node as root. We require that its root is a noun, otherwise we don't consider it a valid instance of causal event.

3. Verb triggers

We can distinguish between two sub-cases. The one with the trigger-verb occurring in active voice:

Noun Phrase (cause) – Verb – Noun Phrase (effect)

and the other in the passive voice:

Noun Phrase (effect) – Verb – Noun Phrase (cause)

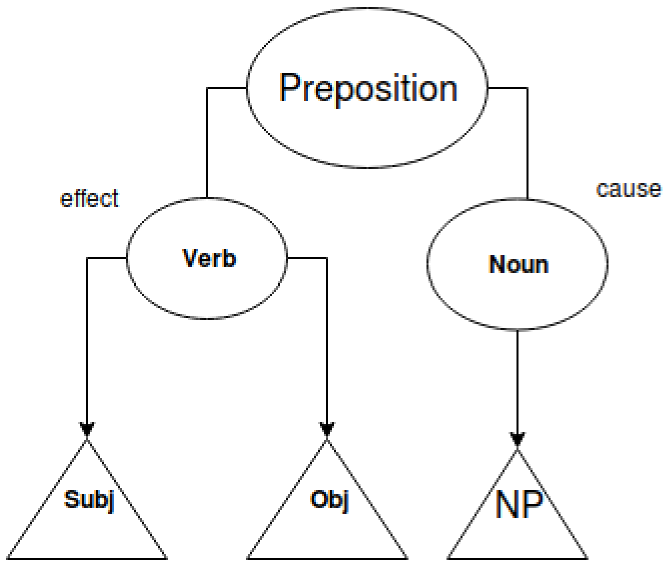


Figure 4.3: Pattern for preposition triggers

The distinction is made by taking under consideration the part-of-speech and dependency tree. More specifically, if the verb is in the past participle form (e.g. allé) and it is not dependent syntactically by an auxiliary verb (être, avoir) then we presume it is occurring in passive voice. Otherwise, it occurs in active voice.

A small detail here is that, although we require the trigger parent node to be a noun we do not have the same strict demand for the child node. We do that because there are many occurrences of causality relation triggered by verbs where the syntactic children of the verbs are stopwords and yet the following syntactically connected words of the corresponding sub-tree constitute a valid causal component, that means that they are good representatives of causal events. It is also important to underline that verb-triggered triplets constitute only a small minority among our extractions.

4.2 Storage of the Collected Data

Extensible Markup Language (XML) is a markup language document encoding through using a format that is both human-readable and machine-readable through use of tags.

The XML file where our extractions are stored, has a very descriptive and detailed form. We aimed at creating a general structured set of causal relation instances in french language. Naturally, it contains some information that didn't find any use in our project. However, we also regard this gathered data as a bequest to other researchers involved in the causality problem in french language.

In our XML file one can find detailed information about each extracted triplet, namely the name of frames triggered by the causal indicator, the indicator's ID, its lemma and

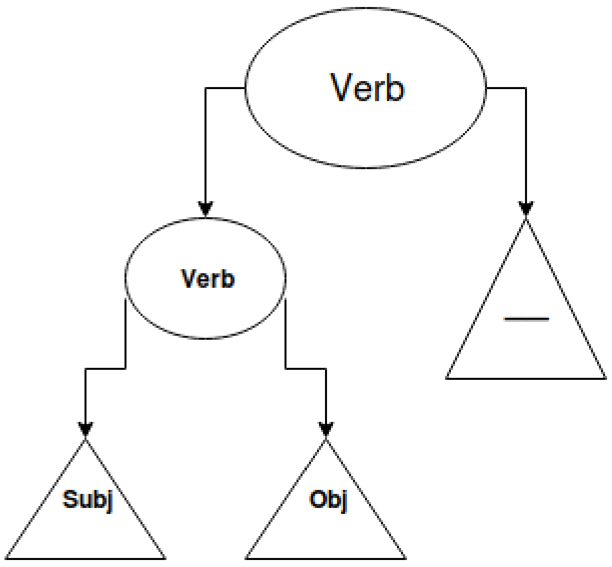


Figure 4.4: Pattern for verb triggers

its form as appeared in the sentence. We also store information about the type of the causal and effect components (Verb Phrase or Noun Phrase). When dealing with a noun phrase we present the part-of-speech, syntactic link, lemma, and natural form of all of its constituents. We act similarly with the Verb Phrase just by additionally discriminating between subject and object trees.

Here is an example of the causal triplet in the xml format, extracted from the sentence "Le sage sait que le nettoyage de l'extérieur n'est pas suffisant, car Dieu voit les profondeurs du coeur, où doit se pratiquer l'ablution du repentir.":

<tuple frame=Causation/Evidence/FR_Cause_enunciation/ id=1402 pos=CC trigger=car word=car>

<effect type=VP>

<Verb lemma=**savoir** pos=V/>

<Subject numOfWords=1 phrase=sage>

<w0 dist=0 lemma=**sage** pos=ADJ synt=suj word=sage/>

</Subject>

<Object1 numOfWords=7 phrase=que est nettoyage n' pas suffisant extérieur>

<w0 dist=0 lemma=**que** pos=CS synt=obj word=que/>

<w1 dist=1 lemma=**être** pos=V synt=obj word=est/>

<w2 dist=2 lemma=**nettoyage** pos=NC synt=suj word=nettoyage/>

<w3 dist=2 lemma=**ne** pos=ADV synt=mod word=n'/>

<w4 dist=2 lemma=**pas** pos=ADV synt=mod word=pas/>

<w5 dist=2 lemma=**suffisant** pos=ADJ synt=ats word=suffisant/>

<w6 dist=4 lemma=**extérieur** pos=NC synt=obj word=extérieur/>

</Object1>

```
    <Object2 numOfWords=0 phrase=/>
  </effect>
  <cause type=VP>
    <Verb lemma=voir pos=V/>
    <Subject numOfWords=1 phrase=Dieu>
      <w0 dist=0 lemma=Dieu pos=NPP synt=suj word=Dieu/>
    </Subject>
      <Object1 numOfWords=6 phrase=profondeurs doit coeur pratiquer ablution
repentir>
      <w0 dist=0 lemma=profondeur pos=NC synt=obj word=profondeurs/>
      <w1 dist=1 lemma=devoir pos=V synt=mod_rel word=doit/>
      <w2 dist=2 lemma=coeur pos=NC synt=obj word=coeur/>
      <w3 dist=2 lemma=pratiquer pos=VINF synt=obj word=pratiquer/>
      <w4 dist=3 lemma=ablution pos=NC synt=obj word=ablution/>
      <w5 dist=5 lemma=repentir pos=NC synt=obj word=repentir/>
    </Object1>
    <Object2 numOfWords=0 phrase=/>
  </cause>
</tuple>
```

4.3 Creation of Causal Embeddings

We used four different training models for the creation of causal embeddings. The first two models were based on Word2vec and the others on SVD and NMF.

4.3.1 Word2vec

The first tool that we employed for this task was word2vec. The word2vec model is normally used for word similarity tasks based on the distributional hypothesis [16]. Usually, the contexts of a word are considered to be words that precede and follow the target word, typically in a window of k tokens to each side. In this project, we chose to use word2vec in a slightly different way by employing arbitrary contexts [23] instead of linear bag-of-words. Till now we had our data stored in the form **cause-trigger – effect triplets** in the xml format as described above, but in this stage, we do away with the triggers which do not anymore give us any useful information and then we use an extra filter so that we create more delicate information in the form of **cause-effect tuples** a set of words that comprises a cause and an effect component serving as our training data set.

Examples of tuples presented in a more succinct form:

Alis à la pensée d' Auguste Terrier, un pas est déjà largement franchi entre ces alter ego, car si le premier peut-être considéré comme l' inventeur du rêve tchadien, le second sera le véritable chef de file des représentations coloniales durables autour_du Tchad.
Cause: second chef véritable file
Effect: pensée ego alter

Engagement qui leur permet, bien entendu, en_même_temps_qu' ils ajoutent leur pierre à l' édifice du Comité de l' Afrique française, de donner du poids à leur pensée, puisqu' il s' agit alors de l' organe de référence sur le sujet.
Cause: organe
Effect: poids pensée

Et cependant, le Bulletin est suffisamment hétéroclite dans sa composition pour_que se pose le problème de la frontière entre ces derniers et ceux qui soutiennent la colonisation du Tchad en_dehors_de ses colonnes <96> c' est notre troisième cercle- mais dont les articles sont repris dans le mensuel.
Cause: problème frontière dernier
Effect: hétéroclite

We used two different training methods in respect to the different forms of the input training data. The first one was a slightly changed implementation of the cEmbed model presented in the paper of Sharp et al. cEmbed is a variant of Mikolov's Skip-Gram with Negative Sampling [26] model called **word2vecf**, implemented by Levy and Goldberg [23] , which modifies the original algorithm to use an arbitrary, rather than linear, context. The novel contribution of Sharp et al.[31] was to make this context task-specific: intuitively, *the context of a cause is its effect.* We followed the same methodology yet without using **word2vecf.** We sticked to the more handy and malleable traditional word2vec implementation in Gensim python library.

In the second method we took the "risk" to do away with the intuitive concept that the cause component is the context of the effect component (and inversly) and we used as contexts all of the words contained in each tuple, both words of the cause and the effect component. Instead of training our model with single cause-effect word-pairs as in the first method, here we create word representations directly correlated with words of the same tuple, indiscriminantly, through the target-context relation. The second method proved to be more fertile than the first one since it provided highest generalization. We should note that in both of these training methods the proper model to be used is Skip-Gram with Negative Sampling and not Hierarchicahal Softmax since we need to exploit both the input and output matrices, something that cannot be done with the latter model.

The **filter** used for the creation of our tuples, either in the form of single cause-effect word-pairs or in the form of cause-effect component-pairs, is crucial in order to minimize noise. Specifically, we dispose of standard french stopwords (e.g. alors, mais, maintenant, ou), words that don't begin with a miniscule letter (such as names, cities, numbers etc.), one-letter tokens and, most importantly, words that are not adjectives, nouns or verbs. We also discarded words that don't belong in a simple french dictionary. Some other filter parameters are important for further calibration of our model's behaviour: we can set a boundary in the depth of the subject, object and noun phrase trees, and we can choose to use only specific frames or use only triplets that are triggered by an indicator of specific part-of-speech. We created more than one training data set by slightly changing various parameters of our filter and evaluating our model's behaviour in each different case.

4.3.1.1 Single word-pair model

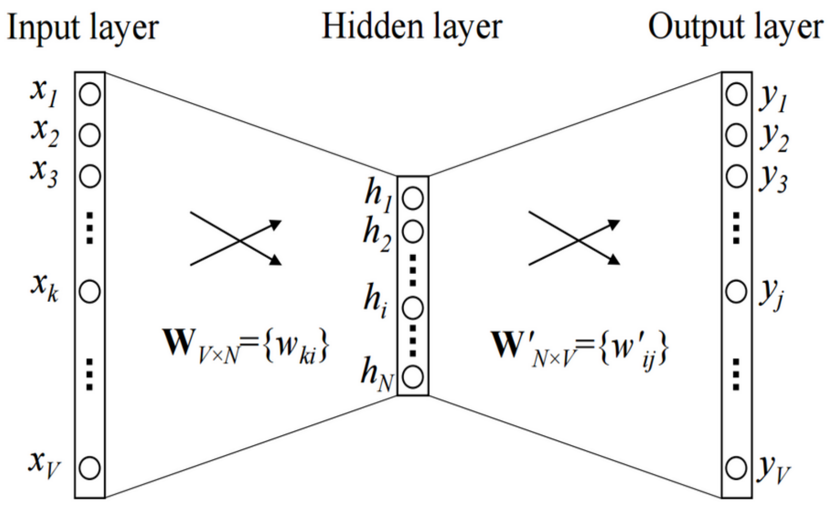


Figure 4.5: Single word-pair model's structure

In this method, our first step was to decompose each cause-effect tuple stored in our xml file, in a way that each word of the cause component is paired with a word in the effect component. From now on, the words of the first kind will be called cause-words and those of the second kind, effect-words. We practically regard these pairs as an input sentence for word2vec. In our case, our input file consists of two columns containing words: a cause column and an effect column. Assuming that the sentences used as training data for word2vec are separated by a change line character. That means that we store our word pairs line by line separated by a space in a simple text file.

An important detail is that we added special prefixes ("cs_" for cause and "ef_" for effect) so that different embeddings can be created for each specific word either by occurring as a cause-word or as an effect-word. The form of the file can be represented like this:

```
cs_word1 ef_word2
cs_word1 ef_word3
```

cs_word4 ef_word5

...

Example of a cause-effect tuple:

cause_component = (second, chef, véritable, file, représentation)
effect_component = (pensée, ego, alter)

and the same tuple decomposed for training purposes:

cs_second ef_pensée
cs_second ef_ego
cs_second ef_alter
cs_chef ef_pensée
cs_chef ef_ego
cs_chef ef_alter
cs_véritable ef_pensée
cs_véritable ef_ego
cs_véritable ef_alter
cs_file ef_pensée
cs_file ef_ego
cs_file ef_alter
cs_représentation ef_pensée
cs_représentation ef_ego
cs_représentation ef_alter

A word in the left column of our data (a cause word) will always have as context a word in the right column (an effect word) and vice versa. We should always keep in mind that a word cs_$word_i$ is different from the word ef_$word_i$.

After the pre-processing, the time comes for us to train our model and create our word embeddings. Similarity between the embeddings can be expressed as the cosine similarity or as the Euclidean dot product (the unnormalized version of the cosine vector proximity).

Cosine Similarity:

$$cos(\theta) = \frac{A \cdot B}{|A|_2 |B|_2} \tag{4.1}$$

It is evident that the cosine similarity measure is the normalized version of the dot product.

What we first of all can expect from our model is to create such vectors for our words so that similar cause-words have similar vectors and that similar effect-words have similar effect-vectors.

That is really the case. We can give a good qualitative example. These are the 10 most "similar" words of the french word cs_**guerre** (death) after training our word2vec with negative_sampling=15, vector_dimensionality=200, syntactic_subtree_depth=12:

cs_napoléonien (napoleonian)	0.741778314114
cs_sanguinaire (bloody)	0.717320024967
cs_pillage (loot)	0.70711171627
cs_anticlérical (anticlerical)	0.706969916821
cs_déchaînement (outburst)	0.706715583801
cs_collatéral (collateral)	0.706022918224
cs_féodalité (feudalism)	0.705951690674
cs_pacifiste (pacifist)	0.705151259899
cs_blindé (tank)	0.70389854908
cs_cataclysme (cataclysm)	0.703700780869

This fact is a very interesting feature of our model's behaviour and it can find some good use in several NLP tasks. Yet, it is not exactly what we were actually aiming for. We will soon examine how this fact gives our model the ability for good generalization. It is also this exact fact that pushed us into testing the second training method of word2vec that we will present later on. Before that though we should get a deeper insight into our initial goals.

4.3.1.1.1 Digression: a better understanding of our objectives

Our main objective is to detect causal relations. For this task, we take advantage of our knowledge about the frequency of the co-occurrence of words as parts of cause and effect components. What was the motive for us to use this specific measure?

Firstly, a conviction, persistent to those of us who use machine learning methods, that patterns that appear in very large set of data will reappear in another. Furthermore, patterns that appeared in the past will reappear in the future.

Our second motive has to do with our adherence (from the point of view of an engineer) to the distributional hypothesis, which is itself justified from the assertion above. This assumption can be re-interpreted in a form, specialized to our task, as the idea that if a cause-effect word pair appears frequently in a big corpus, then the occurrence of these words, relatively close and syntactically connected to each other in a sentence or generally in a text, will indicate a high probability of the existence of causality relation. In a case like this we say that there appears high **causal proximity** between the two words. Notice that we don't rely any more on special causal indicators as indicators of causal relations. The implicit causality relation instances (cases where causality is not triggered by special indicators) are numerous in all of human languages and it is especially this challenge that we are facing when trying to tackle causality detection

4.3.1.1.2 Input and Output Embeddings

Now, it's time to explain how our newly created vectors can indeed be useful for our own task. The idea is based on a statement of Levy and Goldberg [23]:

"[SGNS's] learning procedure is attempting to maximize the dot product v c ·v w for good (w, c) pairs and minimize it for bad ones. "

This assertion can be explained by the formula (3.64) for the loss function of SGNS (Skip-Gram with Negative Sampling). Having this idea in mind, we can choose as a measure of **causal proximity** of two words, the dot product of their corresponding input and output vector. The cosine similarity proved to be a valid measure (and even more accurate), too, something that didn't come as a surprise, since it merely constitutes a normalized version of the dot product.

As a consequence, the resulting trained model of ours gives strongly correlated output and input vectors. The context vocabulary C is identical to the word vocabulary W in contrary to the architecture of word2vecf (the variant model of Levy and Goldberg) where W contains only words of the first column (target-words) and C contains only words of the second column (context words). Furthermore, concerning the input and output matrices, each one corresponds to words of only one event, either cause or effect, depending on the arrangement of the input data. The identifiability of W and C and, henceforth, the direct (one-step) bidirectional training of our model is the only essential difference between our training technique and the one of Sharp et al. In their project Sharp et al. have also employed bidirectional training (cEmbedBi), yet, in two distinct steps by employing word2vecf and they observed ameliorated results. We followed their example in the case of our matrix factorization methods.

It is important to underline here that the special form of our training data, the arbitrary causal context approach that we employed by using word2vec constitutes a novelty of ours. This technique can be used for any kind of context categorization (for whatever number of different categories) just by adding a special suffix to the words of each distinct category.

The strength of our method resides on its **generalization** since, as we noted above, words of the same column (the ones deriving from the same event component) have similar vectors. Even test pairs which have never co-occurred in our training set will be correlated due to the similarity of the first word of the pair with other words of the same event (cause or effect) that have indeed co-occurred with the word of the other event. This phenomenon is exactly what we noted above about the similarity between words of the same event (cause or effect).

The above feature of our model is what renders it suitable for our task in contrast to other distributional methods very often used in the past [10] for causality detection.

4.3.1.2 Tuple-based model

In this method, we don't decompose our filtered tuples into pairs but instead we use the whole tuple intact as an input "sentence" for word2vec. Of course, this time we used a very large context window, at least as large as the number of the longest used tuple. We stuck to the suffix based event discrimination. Thus, the form of a each single input line in the text file derived from our causal xml extractions looks like this:

cs_wordi . . . cs_wordi+p ef_wordj ... ef_wordj+q

The same tuple givean as example above transforms into:

cs_second cs_chef cs_véritable cs_file cs_répresentation ef_pensée ef_ego ef_alter

Once again, after the training phase we use cosine similarity as a measure of causal resemblance between two word embeddings of the input and the output matrices. We will take a good look at the produced results in the next section that concerns our models' evaluation. At this point we restrict ourselves to a mere attempt of explaining the observed improvement of our quantitative measures.

Our interpretation of this phenomenon is based on the idea presented above concerning the generalization achieved through the high correlation among same event-type words. It is this idea that pushed us into employing this second method which is practically different from the previous one, only in that it invests more in the creation of same event-type word correlations. This very interesting feature has unfortunately its own **cost** that we will discuss soon.

With this method it would make sense to use the Embeddings stored in one matrix, naturally the input matrix. Similar contexts give similar vectors right? Yet, our model captures causal proximity much better when we use as a measure the cosine similarity of input and output vectors instead of exploiting only one matrix. This fact makes it evident that this idea about employing knowledge from both input and ouput matrices of word2vec seems to be very interesting and unfortunatelly not much attention has yet been paid to it. The interpretation of such a behaviour still remains to be done.

4.3.2 Matrix Factorization

The second tool we employed to create causal embeddings is matrix factorization; specifically SVD and NMF. For these unsurpervised training methods we use the exact same form of training data as the one used in the single word-pair method for word2vec training. Each cause-effect pair indicates the co-occurrence of the corresponding words in a causality frame instance of our french corpus. Thus, we created a matrix Anxn, where n is our vocabulary size containing the PPMI values for each pair. We then factorized this matrix so that it could be expressed as the inner product of two other matrices: Unxk and Vnxk, where k is a value of our choice (practiaclly around 100 to 200). The rows of these two matrices constitute our new vector representations. Bearing in mind that UxV' is an approximation of the initial A matrix we deduce that the dot product of U[i] and V[j] vectors is an approximation of the cell A[i,j] which contains the value PPMI(U[i], V[j]). It is exactly the value of PMI or any of its variant that is practically the measure of co-occurence probability in respect to the individual frequency of each words and thus it is a very accurate and explicit measure of the **causal proximity** between two words. A small detail is that we didn't use the actual PPMI but the rounding down to integer (int8) of 10*pmi value, so that we avoid the use of high rate memory usage.

A reasonable question that arises is why do we use an approximation of PMI and not the actual function. Why haven't we actually used this measure in the first place instead of employing all these costly training models? The answer is simply that since we need to measure the causal proximity of two words, if we rely on the PMI, it is imperative that our words have indeed co-occured, that we have this word-pair in our training data. This is practically impossible if we consider a vocabulary size of 15,000 to 20,000 words (we are always talking about lemmas). Furthermore, we understand that many of the cells of A matrix are 0.

We used two separate matrix factorization techniques: SVD and NMF. The former performs much better than the latter. SVD also outperforms the single word-pair word2vec model. The relatively satisfying results have to do once again with the high generalization achieved through matrix factorization when using small singular value matrices, that is low dimensional vector representations (around 100-200 dimensions). Th superiority of SVD relies on its proximity with Skip-Gram in terms of mathematical foundation, since as Levy and Goldberg argued, Skip-Gram with Negative Sampling is is implicitly factorizing a word-context matrix, whose cells are the pointwise mutual information (PMI) of the respective word and context pairs, shifted by a global constant.

4.3.3 DSMs vs. Predict models

Both SVD and NMF are Distributional Semantic Models (DSMs). We can view DSMs as count models since they practically count co-occurrences among words by operating on co-occurrence matrices. In contrast, neural word embedding models can be seen as prediction models, since they form their embeddings during the training phase by trying to predict surrounding words. Both DSMs and word embedding models act on the same underlying statistics of the data, i.e. the co-occurrence counts between words.

SVD has some computational advantages over neural embedding models that use SGD such as that it is exact, and does not require learning rates or hyper-parameter tuning. Also, it is trained on count-aggregated data (i.e. {(w, c, #(w, c))} triplets), and hence it can be applied to much larger corpora than SGNS, which requires each observation of (w, c) to be presented separately. [24]

Yet, SGD as well has several advantages. For example, SVD can't cope with unobserved values as efficiently as SGD and SGD can distinguish between observed and unobserved events even though SGD can't [24].

Chapter 5

Evaluation

We begin the testing of our models with a quantitative evaluation of their ability to capture cause-effect relations. We compare their results with a simple baseline. In order to avoid bias towards our extraction methods, we evaluate our models on a partially translated (100 cause-effect pairs and 100 word pairs of other relations) drawn from the SemEval 2010 Task 8 [18], originally a multi-way classification of semantic relations between nominals. SemEval (Semantic Evaluation) is an ongoing series of evaluations of computational semantic analysis systems, organized under the umbrella of SIGLEX, the Special Interest Group on the Lexicon of the Association for Computational Linguistics.

5.1 Test Data

In our test data we have been given pairs of nominals that contain the meaning of the two components that constitute each relation. The full list of our nine relations follows is shown below:

Cause-Effect: An event or object leads to an effect.
Example: Smoking causes cancer.

Instrument-Agency: An agent uses an instrument.
Example: laser printer

Product-Producer: A producer causes a product to exist.
Example: The farmer grows apples.

Content-Container: An object is physically stored in a delineated area of space, the container.
Example: Earth is located in the Milky Way.

Entity-Origin: An entity is coming or is derived from an origin (e.g., position or material).

Example: letters from foreign countries

Entity-Destination: An entity is moving towards a destination.
Example: The boy went to bed.

Component-Whole: An object is a component of a larger whole.
Example: My apartment has a large kitchen.

Member-Collection: A member forms a nonfunctional part of a collection.
Example: There are many trees in the forest.

Communication-Topic: An act of communication,whether written or spoken, is about a topic.
Example: The lecture was about semantics.

There is a tenth element added to this set, the pseudo-relation OTHER. It stands for any relation which is not one of the nine explicitly annotated relations.

Our models rank the pairs using the cosine similarity (for word2vec) and dot product (for SVD and NMF) between cause-vectors and effect-vectors. We aim at ranking the causal pairs above the others.

5.2 Baseline: Vanilla Embedding Model

As a baseline model we used a word2vec model (pre-)trained on the same french corpus (frWac) that we also used to extract our causal triplets. We call this model Vanilla Embedding Model. It is trained on lemmas using 500 dimensional vectors. As with the Single word-pair and the Tuple-based model, SemEval pairs were ranked using the cosine similarity between the vector representations of their arguments.

5.3 Statistical Measures: Precision and Recall

For our evaluation we firstly use a precision-recall curve. The precision-recall curve shows the trade-off between precision and recall for different thresholds. Precision can be seen as a measure of exactness or *quality*, whereas recall is a measure of completeness or *quantity*.

Precision (P) is defined as the number of true positives (T_p) devided by the number of true positives plus the number of false positives (F_p).

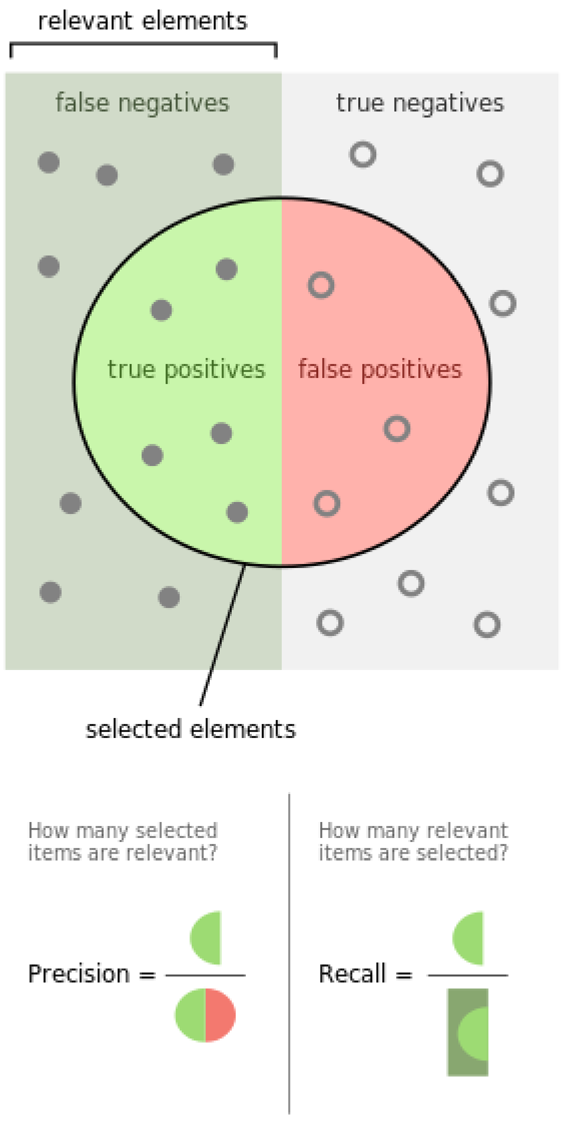


Figure 5.1: Picturing of Precision and Recall on a random classification task

$$P = \frac{T_p}{T_p + F_p} \tag{5.1}$$

Recall (R) is defined as the number of true positives (T_p) devided by number of true positives plus the number of false negatives (F_n).

$$R = \frac{T_p}{T_p + F_n} \tag{5.2}$$

True positives indicate the number of items that are correctly classified as belonging to the positive class. ***False positives*** represent the number of items that were incorrectly classified as belonging to the positive class. ***False negatives*** are the items which were not classified as belonging to the positive class but should have been. ***True negatives*** are the items which were not classified as belonging to the negative class but should have been.

5.4 Results

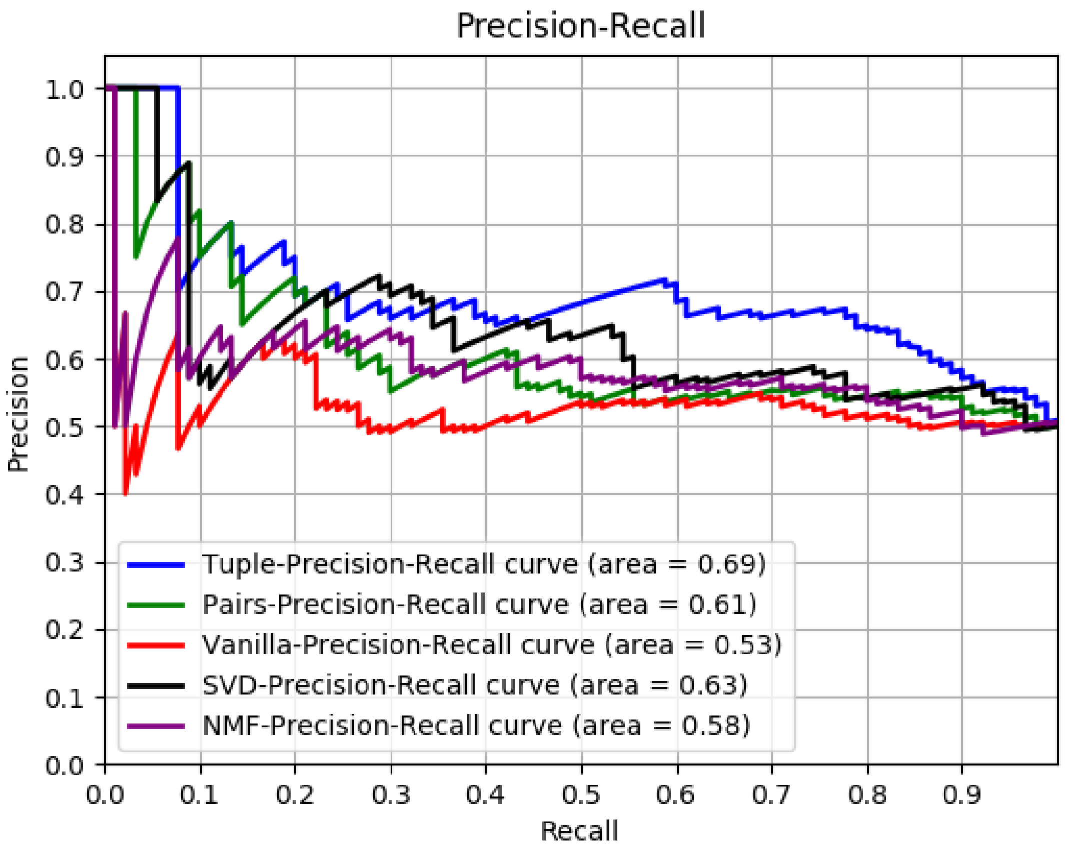


Figure 5.2: Precision-Recall: Our testing results

Figure 5.2 represents the Precision-Recall curve for our 4 causal models and our vanilla one. We chose to compare here our models for the parameters that appeared to ensure the best possible performance for each individual model. We will examine soon how different training and filter parameters result in slight differences in the behavior of our models.

The **Area Under Precision-Recall Curve** (AUC), indicated in the graph simply as area, is obtained by the trapezoidal interpolation of the precision. We use this measure to rank the performance of our models.

The ranking goes as following, from worst to best:

1. Word2Vec Tuple-based model
2. SVD model
3. Word2Vec Single word-pair model
4. NMF model
5. Vanilla Embedding model

As expected, our causal models are much better able to rank causal pairs than our baseline. Incorrect rankings were largely driven by low frequency words whose embeddings could not be robustly estimated due to lack of direct evidence. The Tuple-based model significantly outperforms the other models especially for high recall values. In the second place comes the SVD model which behaves quite well for recall values smaller than 0.5. It is admitedly not as stable as our Tuple-based model. Next comes our Single Word-pair Embedding model with an unsatisfying capacity of detecting the causality frame instances in comparison with the above methods and the Arizona's almost identical model (yet for a different training set, specifically in the English language). Nextly, we can notice the poor predictive behavior of the NMF model. Finally, it becomes evident from the results of the Vanilla Embedding Model its inadequacy for causality detection tasks.

Another representation of the performance of our models can be achieved by using the Receiver Operating Characteristic (ROC). In statistics, a receiver operating characteristic curve, i.e. ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. This graphical representation (Figure 5.3) gives us very similar results with the previous one.

5.5 Discussion: Qualitative Results

At this point we will try to give a better insight into the behavior of our models by looking at a few of our SemEval test pairs and check each model's rating.

The first three are nominals appeared in causal sentences. They are pairs of such kind that it would be easy for a human to deduce a causal link between the two nominals without looking for causal indicators (in this case the verbs "trigger" and cause).

sentence: A <e1>fire</e1> triggered by the <e2>blasts</e2> damaged eight buildings at the plant, including one that was burned down.

Translated test pair: explosion feu

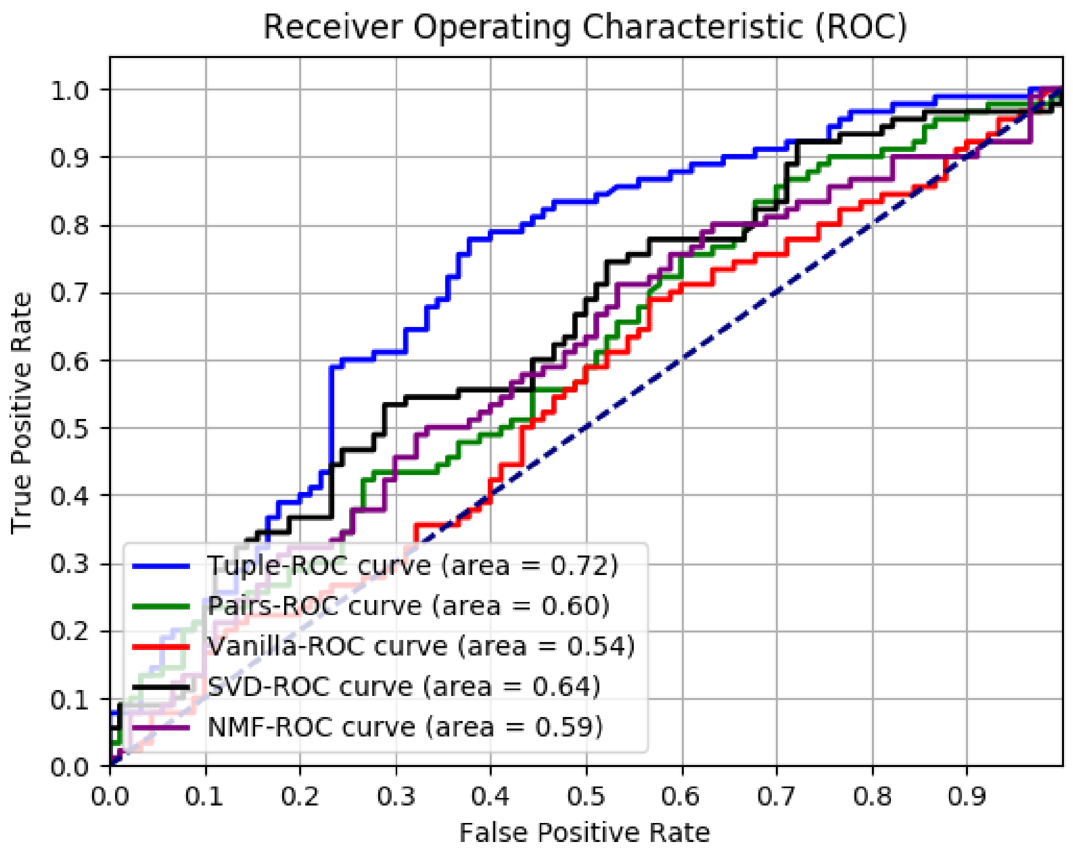


Figure 5.3: Receiver Operating Characteristic: Our testing results

Relation Type: Cause-Effect (e2,e1)
Tuple-based model rating: -0.130009
Single word-pair model rating: -0.0197744
Vanilla Embedding model ratig: 0.364054
SVD model rating: 0.212785
NMF model rating: 0.107053

Sentence: *But the <e1>earthquake</e1> triggered by the <e2>eruption</e2> of Thera struck first.*
Translated test pair: séisme éruption
Relation Type: Cause-Effect (e2,e1)
Tuple-based model rating: -0.0908807
Single word-pair model rating: 0.0491519
Vanilla Embedding model ratig: 0.473948
SVD model rating: 0.207729
NMF model rating: 0.222042

Sentence: Once they grow there, the <e1>swelling</e1> and inflammation caused by the <e2>infection</e2> closes off the sac, causing it not to shed bacteria, and protecting the bacteria inside from antibiotics and your body's own immune cells.

Tranlated test pair: infection inflammation

Relation Type: Cause-Effect (e2,e1)

Tuple-based model rating: -0.0287006

Single word-pair model rating: 0.0601631

Vanilla Embedding model rating: 0.502571

SVD model rating: 0.523963

NMF model rating: 0.445636

The next three are pairs of other kinds of relations, for which a human would admittedly give a low probability of designating causal relation.

Sentence: <e1>Beaver</e1> <e2>dams</e2> are created as a protection against predators

Translated tuples: castor barrage

Relation Type: Product-Producer(e2,e1)

Tuple-based model rating: -0.416389

Single word-pair model rating: -0.336954

Vanilla Embedding model ratig: 0.30219

SVD model rating: 0.143796

NMF model rating: 0.0290136

Sentence: There has been intense <e1>debate</e1> over the circles' <e2>origins</e2>.

Tranlated test pair: débat origine

Relation Type: Message-Topic(e1,e2)

Tuple-based model rating: -0.384409

Single word-pair model rating: -0.132343

Vanilla Embedding model rating: 0.12804

SVD model rating: -0.0343494

NMF model rating: 0.029079

Sentence: The second <e1>simulation</e1> was started from the uncomplexed x-ray <e2>structure</e2> after insertion of the ligand into the binding site.

Translated test pair: simulation structure

Relation Type: Entity-Origin (e1,e2)

Tuple-based model rating: -0.21932

Single word-pair model rating: -0.0565504

Vanilla Embedding model rating: 0.325763

SVD model rating: 0.100709

NMF model rating: 0.0470929

In a few words, in these six examples it is easy for humans to detect or deny causality. We can see that actually our models perform relatively well considering that they give much smaller ratings to non-causal pairs. Note that the rating values of our model cannot be compared among them. Word2vec seems to be much more "strict" than our factorization models. The low ratings given by the Tuple -based method can be partly explained by a special feature of it that we haven't yet discussed. It is the fact that the causal proximity gets mixed up with word similarity in terms of causal contexts. To make this plain clear we can give as example the results of the top 10 closest (most similar output embeddings for input embedding of the word (cs_)**guerre** (war):

cs_mondial: 0.00306198
cs_civil: -0.00311063
ef_saoudite: -0.0331004
cs_meurtrier: -0.033403
cs_atrocité: -0.0369734
ef_syrien: -0.0421228
cs_sanglant: -0.0468483
cs_décolonisation: -0.0505369
cs_invalidité: -0.0521121
cs_atroce: -0.0541832
cs_terrorisme: -0.054809

Here, we see that the majority among the top 10 "most similar" words are cause-words, words of the same event type as the word geurre. This attribute of our model does not constitute any problem as far as the event types of the words in our vocabulary can be distinguished by their prefix. These results also show us that our model is indiscriminantly "strict" giving negative cosine values almost for every possible word-pair.

The behavior of our Single word-pair model is different, and even though it seems to work more robustly, the quantitave results above disprove this almost as an illusion. Once a gain we can check the top ten results among output embeddings for the word cs_guerre of the input matrix:

ef_guerre: 0.117155
ef_destruction: 0.0765318
ef_saoudite: 0.0642342
ef_massacre: 0.0558169
ef_nazi: 0.0459338
ef_armée: 0.0444262
ef_ravage: 0.0413298
ef_militaire: 0.0381308
ef_empire: 0.037602

ef_paix: 0.0344615
ef_mort: 0.0330351
ef_soldat: 0.0304276

5.5.1 Interpretation of our Models' Flaws

Here we will try to give an interpretation of our models' flaws by examining some of the difficulties that they are confronting in this specific evaluation task.

Many flaws originate from the phase of the training set creation. One major problem is that, if a word is part of a causal component in the training tuples, it does not really mean that it is the main bearer of the meaning of the component (the actual causal event). It is really difficult to create such a filter that can cope with such noise. This constitutes the most important factor of our problem. The main source of our inefficacy.

A relevant example is this pair of words, heel and shoe, that could naturally appear in causal components simply because of their close relation and thus their possible syntactic link in a sentence of our corpus.

Sentence: He decided to pad the <e1>heel</e1> of <e2>shoes</e2> with a shock absorbing insole or heel pad.
Tranlated test pair: talon chaussure
Relation Type: Other
Tuple-based model rating: -0.0290125
Single word-pair model rating: -0.0525966
Vanilla Embedding model rating: 0.577776
SVD model rating: 0.132815
NMF model rating: 0.2837

Another problem is the orthography and types of mistakes in the frWac corpus. Also, the imperfections of the annotation program play a major role. The combination of these two flaws create serious blemishes to our training data.

Other problems have their origin in the difficulties that we have to face specifically when testing our models with SemEval pairs. For example, even humans wouldn't expect high causal proximity among test pairs like the ones in the next examples:

Sentence: The <e1>disruption</e1> has been caused by a sensitivity reaction in the brain to an ingested <e2>substance</e2>.
Tranlated test pair: perturbation substance
Relation Type: Cause-Effect

Tuple-based model rating: -0.242246
Single word-pair model rating: -0.118238
Vanilla Embedding model rating: 0.199395
SVD model rating: 0.129461
NMF model rating: 0.0699259

Sometimes human rating of the causal proximity might depend on our beliefs, our ideologies. In terms of pattern recognition, what corresponds to the human beliefs is the ideological inclinations of the sources from which originated our training set, that is our corpus. If we want to build a general purpose machine we have to expect to face certain difficulties, of the same type that a totally non-biased human-being would face. Let's take a glimpse into the following SemEval example:

Sentence: police officials offer apologies for the <e1>suffering</e1> caused by the responsible police <e2>officers</e2>.
Tranlated test pair: policier souffrance
Relation Type: Cause-Effect (e2, e1)
Tuple-based model rating: -0.293355
Single word-pair model rating: -0.0570472
Vanilla Embedding model rating: 0.181807
SVD model rating: 0.115398
NMF model rating: 0.332066

Here all of our models predict relatively low causal proximity except of our NMF that we can assume that its response is quite arbitrary since taking under consideration the above quantitative results. This example made apparent one more difficult that our model has to cope with, but at the same time implicitly it gives us some information about the corpus used for training and hence an insight on the ideological contents appearing on the internet.

Another difficulty is that a very common word may appear much more often in causal relations in our corpus together with specific words and not with some others even if the last ones have obvious causal proximity with it. For example, the word price in our next test pair, where the collapse is a perfectly possible effect yet a very special one:

Sentence: The low oil <e1>prices</e1> caused the <e2>collapse</e2> of the wall and the Russian empire.
Tranlated test pair: prix effondrement
Relation Type: Cause-Effect (e1, e2)
Tuple-based model rating: -0.312424
Single word-pair model rating: -0.165238
Vanilla Embedding model rating: 0.12077

SVD model rating: 0.0568018
NMF model rating: 0.150254

Another good example of the same type is the following involving the word movement:

Sentence: The beautiful hydrothermal features in the park (geysers, hot springs, mud pots, etc.), the uplift and subsidence, and many of the <e1>earthquakes</e1> are caused by the <e2>movements</e2> of hydrothermal and/or magmatic fluids.
Tranlated test pair: séisme mouvement
Relation Type: Cause-Effect (e2, e1)
Tuple-based model rating: -0.196879
Single word-pair model rating: 0.00350669
Vanilla Embedding model rating: 0.196323
SVD model rating: 0.16437
NMF model rating: 0.0426388

Other causes of our models' inefficacies would include imperfections in our pairs' translation. We can think that a translation from single word to single word is not an easy task and can cause differentiations in the meaning of our test corpus.

5.6 Extra Quantitative Tests

We also tested weather bidirectionality could imporve our evaluation results in the case of SVD as [31] had proposed for the case of the word2vecf model. We should note here that with our word2vec models bidirectionality is ensured in the first place and thus we don't need a second training phase.

The results do not show any major improvement. We could though notice a slight amelioration of our model's stability.

Finally, we present some other tests that show how different parameters of our Tuple-based training model have an effect on the behaviour of our model. These graphs represent also more or less the procedure we followed in order to conclude in our final choice of parameters.

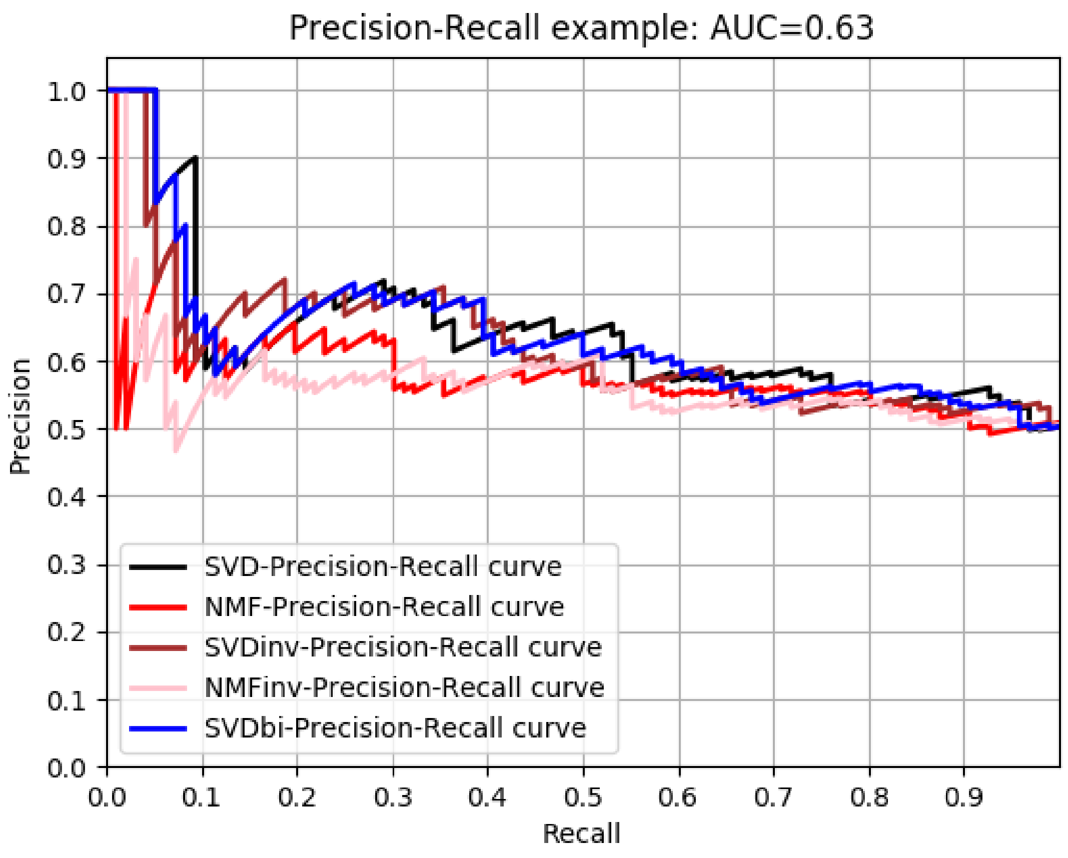


Figure 5.4: Precision-Recall: Bidirectional SVD

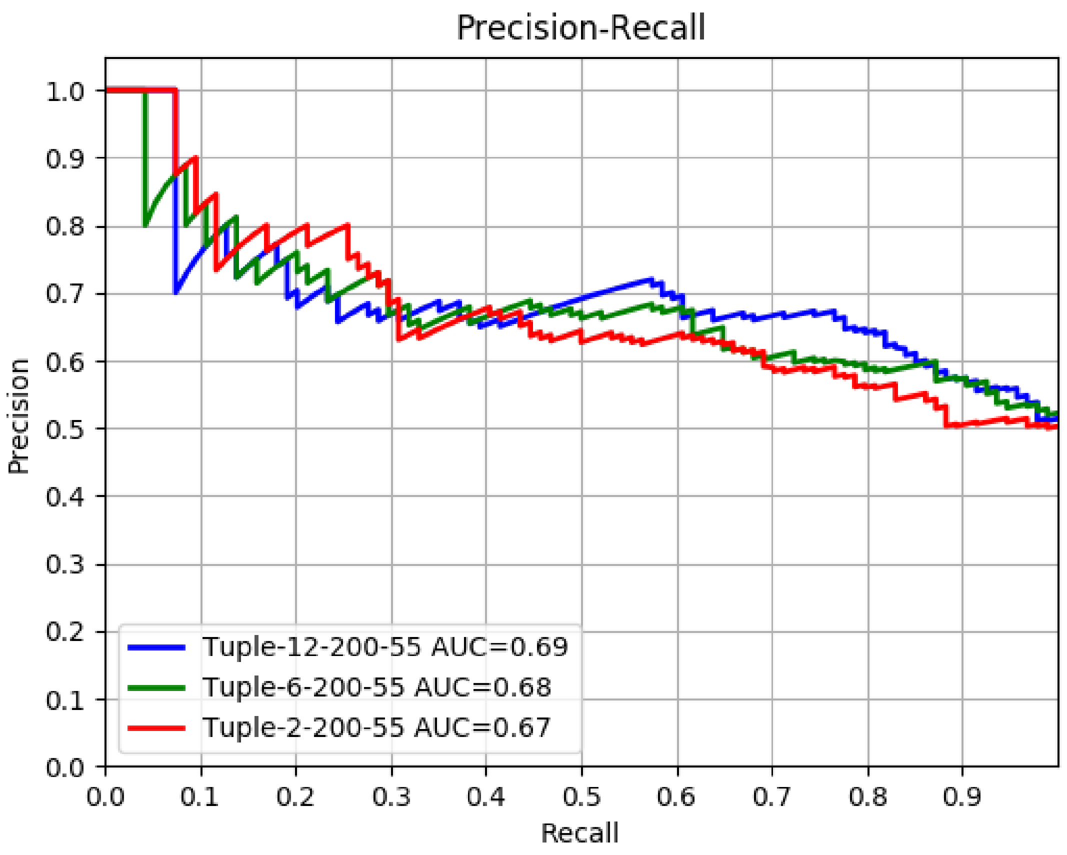


Figure 5.5: Precision-Recall: Impact of different dependency subtrees depth on the Tuple-based model

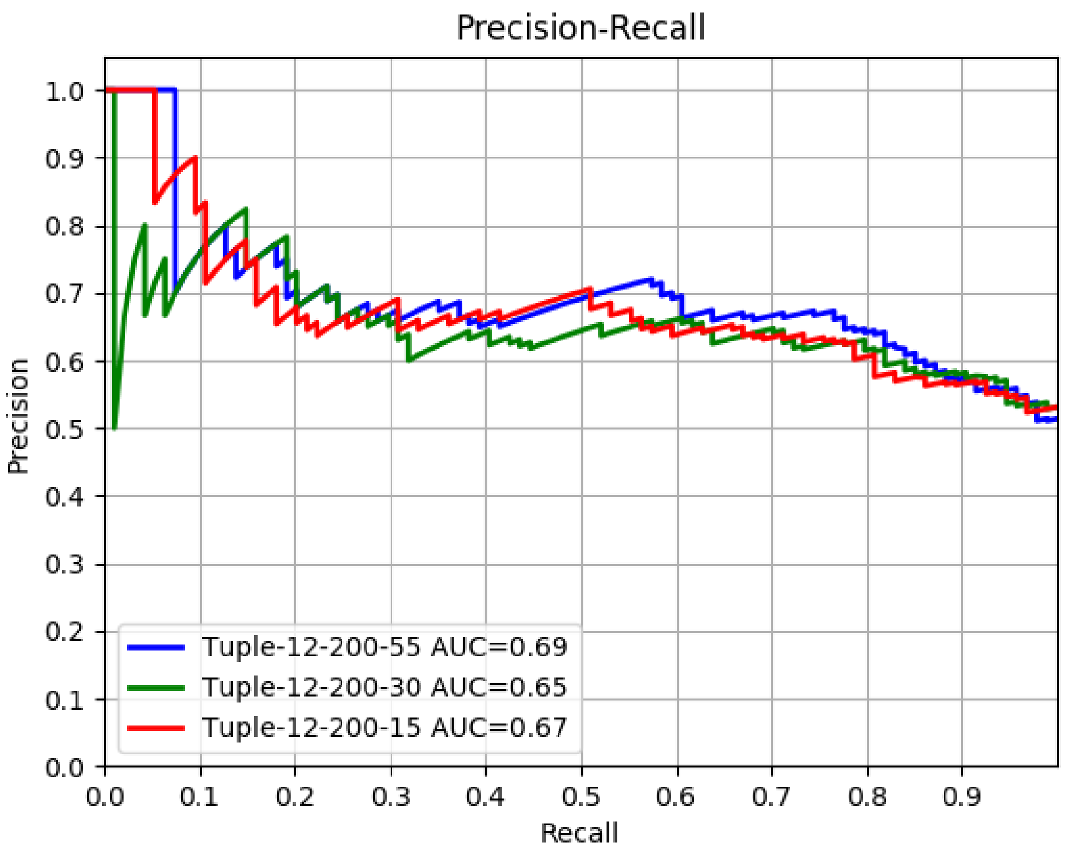


Figure 5.6: Precision-Recall: Impact of different number of negative samples on the Tuple-based model

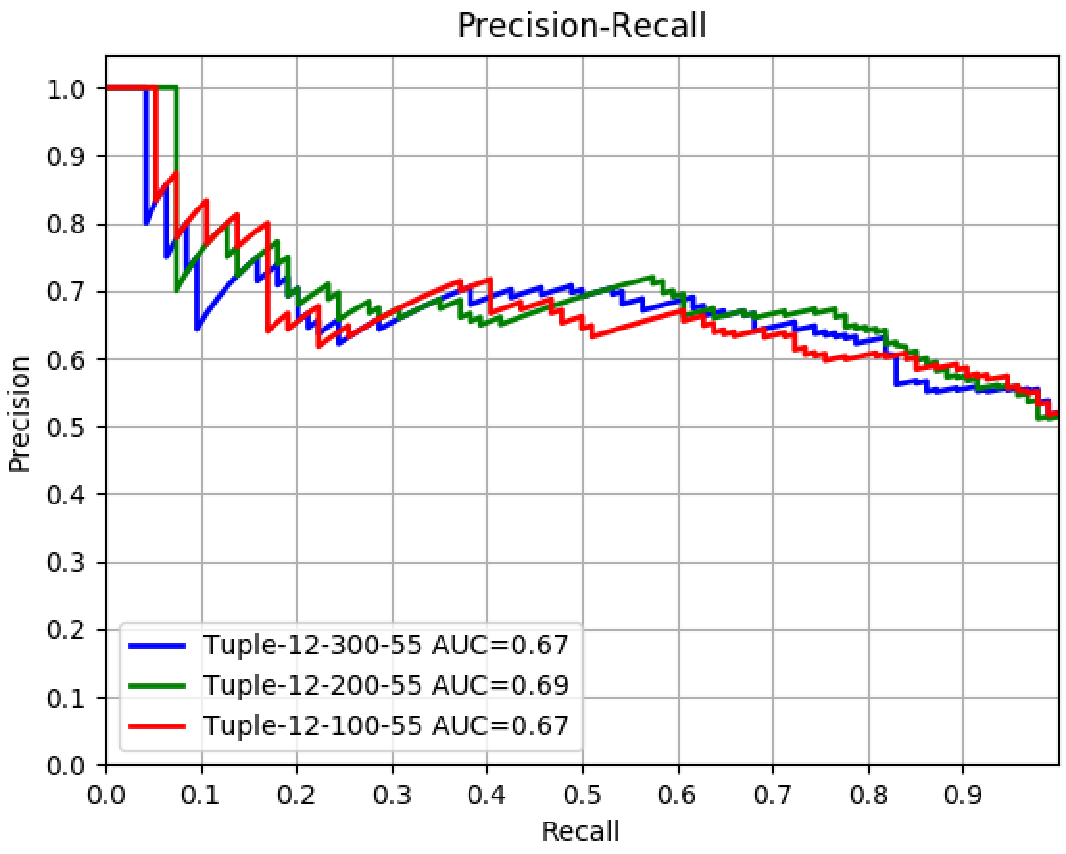


Figure 5.7: Precision-Recall: Impact of dimensionality on the Tuple-based model

Chapter 6

Conclusion

Finally, we have been able to draw some conclusions. After the comparison of our models' behaviour and the deep understanding of their inner structure and functioning it has become clear to us that the ability of generalization plays the most crucial role in the effectiveness of our models. This fact has become clear also by the inadequacies of the PMI measure to cover our needs and by the apparent superiority of our Tuple-based model in relation to the others.

We consider important to underline the fact that our models face some difficulties in causality detection tasks where there appear highly complex sentences since, in our project, many of the words that form their meaning, were seen merely as stop-words. We actually believe that many of our models' flows are related with the specificities of our training sets. Choosing the proper training data is a great challenge for relation extraction in general. We think that there is still a lot of work to be done pertinent to the process of automated collection of good causality instances.

Bibliography

[1] N. Asghar. Automatic extraction of causal relations from natural language texts: A comprehensive survey. *arXiv preprint arXiv:1605.07895*, 2016.

[2] S. Banerjee and A. Roy. *Linear algebra and matrix analysis for statistics.* Chapman & Hall/CRC texts in statistical science series. CRC Press, Taylor & Francis Group, Boca Raton, 2014.

[3] M. Baroni, S. Bernardini, A. Ferraresi, and E. Zanchetta. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226, 2009.

[4] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[5] C. M. Bishop. *Pattern recognition and machine learning.* Information science and statistics. Springer, New York, 2006.

[6] G. Bouma. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, pages 31–40, 2009.

[7] A. Budanitsky and G. Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47, 2006.

[8] J. A. Bullinaria and J. P. Levy. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods*, 39(3):510–526, 2007.

[9] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[10] J. Conrath. *Unsupervised extraction of semantic relations using discourse information.* PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier, 2015.

[11] J. R. Firth. A synopsis of linguistic theory 1930-55. 1952-59:1–32, 1957.

[12] D. Garcia. Coatis, an nlp system to locate expressions of actions connected by causality links. *Knowledge Acquisition, Modeling and Management*, pages 347–352, 1997.

[13] R. Girju. Automatic detection of causal relations for question answering. In *Proceedings of the ACL 2003 workshop on Multilingual summarization and question answering-Volume 12*, pages 76–83. Association for Computational Linguistics, 2003.

[14] Y. Goldberg. A primer on neural network models for natural language processing. *J. Artif. Intell. Res.(JAIR)*, 57:345–420, 2016.

[15] G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224, 1965.

[16] Z. Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

[17] Z. Harris. *Mathematical structures of language*. Interscience tracts in pure and applied mathematics. Interscience Publishers, 1968.

[18] I. Hendrickx, S. N. Kim, Z. Kozareva, P. Nakov, D. O. Séaghdha, S. Padó, M. Pennacchiotti, L. Romano, and S. Szpakowicz. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 33–38, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[19] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[20] I. T. Jolliffe. *Principal component analysis*. Springer series in statistics. Springer, New York, 2nd ed edition, 2002.

[21] C. S. Khoo, S. Chan, and Y. Niu. Extracting causal knowledge from a medical database using graphical patterns. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 336–343. Association for Computational Linguistics, 2000.

[22] C. S. Khoo, J. Kornfilt, R. N. Oddy, and S. H. Myaeng. Automatic extraction of cause-effect information from newspaper text without knowledge-based inferencing. *Literary and Linguistic Computing*, 13(4):177–186, 1998.

[23] O. Levy and Y. Goldberg. Dependency-based word embeddings. In *ACL (2)*, pages 302–308, 2014.

[24] O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

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

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

[27] A. Ng. Lecture notes 1. CS 229: Machine learning. Technical report, Stanford, CA, 2003.

[28] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation.

[29] S. Ruder. On word embeddings - part 1. `http://ruder.io/word-embeddings-1/index.html`, 2016.

[30] S. Ruder. On word embeddings - part 2: Approximating the softmax. `http://ruder.io/word-embeddings-softmax/index.html#hierarchicalsoftmax`, 2016.

[31] R. Sharp, M. Surdeanu, P. Jansen, P. Clark, and M. Hammond. Creating causal embeddings for question answering with minimal supervision. *arXiv preprint arXiv:1609.08097*, 2016.

[32] L. N. Trefethen and D. Bau III. Numerical linear algebra. philadelphia: Society for industrial and applied mathematics. Technical report, ISBN 978-0-89871-361-9, 1997.