Models and algorithms for automatic text simplification

El Mehdi Issouani

Université Paris Nanterre
200 av. de la République
92001 Nanterre Cedex
www.parisnanterre.fr

Membre de l'université Paris Lumières

Connaissance, Langage, Modélisation (CLM) – ED 139
MODAL'X – UMR 9023

El Mehdi Issouani

Modèles et algorithmes de simplification automatique de textes

Thèse présentée et soutenue publiquement le 23/06/2023
en vue de l'obtention du doctorat de Mathématiques appliquées et applications des mathématiques de l'Université Paris Nanterre
sous la direction de M. Patrice Bertail (Université Paris Nanterre)

Jury:

Rapporteur :	M. Amor Keziou	MCF (HDR), Université de Reims Champagne-Ardenne
Rapporteure :	Mme Estelle Kuhn	DR (HDR), INRAE (Jouy-en-Josas)
Membre du jury :	M. Antoine Chambaz	PR, MAP5, Université Paris Cité
Membre du jury :	Mme Delphine Battistelli	PR, MODYCO, Université Paris Nanterre
Membre du jury :	M. Jean-François Pradat-Peyre	PR, Université Paris Nanterre
Membre du jury :	Mme Marianne Clausel	PR, IECL - Université de Lorraine

Remerciements

Je tiens tout d'abord à exprimer ma profonde gratitude envers mon directeur de thèse, Patrice Bertail, pour sa confiance et pour m'avoir proposé ce sujet passionnant, ainsi qu'envers Emmanuelle Gautherat, avec qui j'ai eu le plaisir de collaborer. Patrice, je te remercie pour ton enthousiasme, ta gentillesse, ton recul et ta perspective éclairée, qui continuent de m'impressionner. Je te suis reconnaissant de m'avoir enseigné à structurer mes idées, à développer une vision d'ensemble et de m'avoir prodigué des conseils avisés qui se reflètent largement dans le contenu et la structure de la version finale de ma thèse. Emmanuelle, je te remercie pour ton écoute scientifique exceptionnelle et ta bienveillance constante. Je te suis reconnaissant d'avoir relu à maintes reprises mes travaux et d'avoir consacré du temps à vérifier la cohérence de mes écrits avec une exigence de clarté et de rigueur qui m'a permis de faire d'importants progrès dans la rédaction de mes résultats. Je vous suis infiniment reconnaissant à tous les deux, non seulement pour votre soutien face à mes problèmes administratifs, mais aussi pour vos encouragements constants et pour tout ce que vous m'avez appris.

Je souhaite adresser mes chaleureux remerciements à Amor Keziou et Estelle Kuhn d'avoir accepté de rapporter ma thèse et d'avoir consacré une attention particulière à la relecture de mon manuscrit. Je remercie vivement Delphine Battistelli, Antoine Chambaz, Marianne Clausel et Jean-François Pradat-Peyre, qui m'honorent en tant que membres du jury.

Je tiens à exprimer ma sincère gratitude envers les chercheurs qui m'ont toujours soutenu et aidé tout au long de mon parcours, tels que Thierry Dumont, Ana Karina Fermin et Mélanie Zetlaoui mais aussi Olivier Baude et François Delbot.

Un grand merci aux membres du laboratoire MODAL'X, où j'ai eu le plaisir et la passion de découvrir la recherche. Je voudrais particulièrement remercier Olivier Couronné, Yann Demichel, Cécile Durot, Nathanael Enriquez, Xavier Mary, Olivier Raimond et Philippe Soulier qui ont été mes enseignants pendant ma licence et mon Master. Ils ont non seulement été mes mentors, me permettant d'acquérir toutes les bases nécessaires pour entamer une carrière de chercheur, mais aussi des collègues très sympathiques avec qui j'ai toujours apprécié de discuter autour d'un café. Je remercie également Cécile Hardouin, Emilie Lebarbier, Christian Léonard et Laurent Mesnager qui m'ont transmis la joie d'enseigner et m'ont permis d'acquérir de solides connaissances en matière d'enseignement et de pédagogie auprès des étudiants. Enfin, je tiens à remercier Olivier Collier, Sylvia Dobyinsky, Xiao Yijun et Nicolas Rauwel pour leurs encouragements fréquents, ainsi que Hanene Mohamed et Cyril Roberto, ainsi que les nouveaux membres tels que Nicolas Marie, Marie Théret et Niccolo Torri, qui m'ont donné envie de poursuivre une carrière universitaire. Je n'oublie pas mes collègues de bureau, doctorants et ATER, avec qui j'ai eu le plaisir de travailler dans une ambiance chaleureuse et conviviale, et qui m'ont toujours apporté leur soutien, notamment Mokhtar, Carlos, Cristina, Karolina, Ali, Amélie, Nathan, Charles, Miraine, Paul, Boutheina, Oussama, Amine, Marwa, Elie, Branda, Hamza, Mohamed Ali et les deux Arij.

Je suis extrêmement reconnaissant envers tous les enseignants exceptionnels qui ont accompagné mon parcours, de l'école maternelle jusqu'au Master. En particulier, je remercie avec une grande émotion

Nicolas De Granrut, Gérard Debeaumarché et François Metayer, qui m'ont fait découvrir la beauté des mathématiques.

Je souhaite exprimer ma gratitude envers tous mes proches et ceux qui comptent dans ma vie, qui m'ont apporté un soutien quotidien précieux, tels que David, Tamer, Walid, Loubna, Hamza, Sofia, Malak, Zakaria, Inès, Ali, Oussama, Doha, Sara, Esther, Samir, Armelle, Léa, Larry et Victorien.

Enfin, je souhaite adresser ma joie et mes remerciements à ma famille : mes merveilleuses sœurs et mon frère, ainsi que leurs enfants, mes nièces et neveux, pour leur soutien inconditionnel tout au long de cette thèse, leur présence et leurs précieux conseils. Mon dernier remerciement est bien sûr pour mes parents. Quelle chance d'avoir grandi à vos côtés, vous qui êtes si formidables, aimants, encourageants et toujours présents. Je vous adresse mes sincères remerciements pour m'avoir aidé à surmonter les périodes d'incertitude que j'ai traversées tout au long de ma thèse. Je suis très fier de vous avoir à mes côtés.

Résumé

Le but de la thèse est de contribuer aux méthodes de simplification automatique de texte. Plus précisément, il s'agit de construire des mesures de complexité (classifieur binaire de textes simples vs textes complexes) et de contribuer au développement de modèles de langage prédictif (comme ceux utilisés dans le désormais célèbre chatbot ChatGPT).

Pour ce faire, nous nous sommes intéressés aux méthodes d'entropie utilisées en NLP (que nous réinterprétons en terme de vraisemblance empirique généralisée) et au comportement de la statistique de Hotelling en grande dimension. Cette statistique apparait naturellement dans ce type de problème et permet d'effectuer des tests de moyenne en grande dimension (c'est-à-dire lorsque la dimension des paramètres dépasse le nombre d'observations). Nous obtenons des bornes exponentielles pour le T^2 de Hotelling, sous des hypothèses faibles de moment. Pour cela, nous proposons un estimateur pénalisé et un choix optimal pour le coefficient de pénalité. Nous discutons également brièvement des modèles de réseaux de neurones dans le domaine du traitement automatique du langage naturel. Nous résumons ensuite les performances d'une série d'architectures d'apprentissage profond sur un corpus Wikipédia pour la construction d'une mesure de complexité et pour la simplification automatique de textes.

L'objectif de ce projet transversal de mathématiques appliquées et de linguistique est d'apporter un éclairage aux problèmes de simplification automatique de texte, ce qui potentiellement pourrait à termes aider les personnes ayant une déficience auditive à faire face aux difficultés rencontrées. Les travaux sont situés dans des champs variés (statistique, linguistique, informatique) et revêt un caractère fortement pluridisciplinaire.

Abstract

The goal of this thesis is to contribute to automatic text simplification methods. Specifically, the objective is to build complexity measures (binary classifiers for simple vs. complex texts) and contribute to the development of predictive language models (similar to those used in the well-known chatbot ChatGPT).

To achieve this, we focus on entropy methods used in NLP, which we reinterpret in terms of generalized empirical likelihood, and the behavior of Hotelling's statistic in high-dimensional settings. This statistic naturally arises in such problems and enables mean testing in high dimensions (when the dimension of the parameters exceeds the number of observations). We obtain exponential bounds for Hotelling's T^2, under weak moment assumptions. For this, we propose a penalized estimator and an optimal choice for the penalty coefficient. We also briefly discuss neural network models in the NLP domain. Then we summarize the performance of a series of deep learning architectures on a Wikipedia corpus for the construction of a complexity measure and an automatic text simplification tool.

This interdisciplinary project in applied mathematics and linguistics aims to shed light on automatic text simplification problems, potentially assisting individuals with hearing impairments in overcoming difficulties. The work covers various fields (statistics, linguistics, computer science) and possesses a highly multidisciplinary nature.

Contents

Introduction

Le phénomène de la surdité à la naissance ou en bas âge pose des questions fondamentales sur l'accès à la langue, notamment via les expériences d'interaction précoce et le développement des compétences en lecture et écriture. Selon une étude menée par Swanwick et al. (2005) [185], la plupart des enfants sourds ont des difficultés de lecture en raison de leur déficit expérientiel et linguistique au début de leur vie, avec un développement cognitif et des compétences linguistiques insuffisantes (voir Quigley et Paul (1984) [162], Quigley (1984) [160], Marschark and Spencer (2010) [132])). Kelly et al. (1996) [100], et Alegria (2004) [2] ont signalé que les lecteurs sourds ont du mal à tirer pleinement parti de leur vocabulaire jusqu'à ce qu'ils atteignent un niveau de compétence syntaxique adéquat à des âges plus avancés. De plus, il a été constaté que ce déficit entre enfants sourds et entendants se creuse au fil du temps (Harris (1994) [87] et Mahapatra (2016) [128]). Des études ont montré que les constructions passives, les propositions relatives, les conjonctions et les pronoms sont des éléments qui affectent la compréhension (voir Robbins and Hatcher (1981) [167]).

Conrad (1979) [50] a montré que le niveau de lecture médian pour l'ensemble d'une population de malentendants (spécifiquement, tous les élèves quittant une école spéciale pour enfants malentendants en Angleterre et au Pays de Galles entre 1974 et 1976) était celui d'un enfant de 9 ans. Pour les personnes présentant une perte auditive supérieure à 86 dB, environ 50% des élèves étaient totalement illettrés. Enfin, même si l'on suppose qu'un niveau fonctionnel de compréhension n'est pas atteint avant un âge de lecture[1] de 11 à 12 ans, moins de 15% de cette population atteignait ce niveau. De plus, un niveau limite de lecture est atteint vers la troisième année d'apprentissage de la lecture (voir également Paul et Jackson (1994) [154], Alegria (2004) [2] et Quigley et al. (1977) [161]). Au final, à tout âge, les personnes ayant une surdité sévère depuis la naissance ou dès leur jeune âge ont souvent des problèmes de lecture et de compréhension des textes.

Dans de nombreux cas, la syntaxe des sites Web, y compris les sites administratifs pourtant indispensables aux citoyens est inadaptée au public souffrant de déficience auditive précoce. Par ailleurs, les services d'assistance vocaux ne peuvent suppléer à cette inadaptation des sites web car ils sont, eux aussi, par nature, inadaptés à ce public. Pour s'en convaincre on regardera la vidéo YouTube[2] où la responsable de l'entreprise DALiNK, Lynda Robillard, simule un appel à un opérateur téléphonique (ou dans d'autres conférences, un appel aux impôts, à des services administratifs ou à Enedis). Selon une enquête OMS[3] de 2021, près d'un million d'enfants naissent chaque année avec une surdité invalidante. En France, 6% des 15-24 ans sont concernés par le déficit auditif invalidant (perte d'audition supérieure à 40 dB pour un adulte et 30 dB pour un enfant). Contrairement à la déficience visuelle, ce problème est mal diagnostiqué, mal corrigé. Pourtant, la déficience auditive a des répercussions importantes sur la vie quotidienne.

Les salons de "chat" en langue des signes [4] soit n'existent pas, soit ne sont pas suffisamment nombreux

[1] L'âge de lecture désigne le niveau de lecture ou de compréhension de la lecture d'une personne sans déficience (surdité, aphasie, dyslexie, etc.). Un âge de lecture de 8 ans représente le niveau de lecture moyen d'enfants de 8 ans n'ayant aucune déficience.

[2] https://www.youtube.com/watch?v=2qLYbVn_ehU

[3] https://www.who.int/fr/news-room/fact-sheets/detail/deafness-and-hearing-loss

[4] L'État a développé un nouvel outil, ANAE, qui est un assistant numérique d'accessibilité. Il est désormais disponible

pour la population concernée ou encore limités à certaines plages horaires. À ce jour, les acteurs du web ne sont pas encore tous conscients de ce type de problème.

Initialement, le but de cette thèse était de contribuer en partenariat avec la startup DALiNK[5] à la simplification de sites web à l'usage des malentendants, afin de garantir un accès équitable à l'information pour les personnes sourdes et malentendantes. Toutefois, il est apparu rapidement qu'il n'existait aucun corpus adapté construit avec des personnes malentendantes (ou trop insuffisant pour être utilisé). Aussi nous n'avons pas traité directement du processus d'adaptation à la surdité, ni d'un point de vue théorique, ni d'un point de vue appliqué. Nous nous sommes plutôt penchés sur le problème de la simplification automatique de texte dans un cadre général incluant les personnes entendantes et malentendantes. Cela inclut donc celles ayant d'autres problèmes impactant leur niveau de lecture ou de compréhension de la langue (dus à divers retards, aphasie, dyslexie etc.) et les locuteurs non natifs d'une langue.

Ainsi, le but de la thèse est de contribuer aux méthodes de simplification automatique du texte. Plus précisément, il s'agit de construire des mesures de complexité (classifieur binaire de textes simples vs textes complexes) et de contribuer au développement de modèles de langage prédictif (comme ceux utilisés dans le désormais célèbre chatbot ChatGPT[6]). Pour ce faire, nous nous sommes intéressés aux méthodes d'entropie utilisées en traitement automatique du langage naturel (TALN), que nous réinterprétons en terme de vraisemblance empirique généralisée et au comportement de la statistique de Hotelling en grande dimension. Cette statistique apparaît naturellement dans ce type de problème et permet d'effectuer des tests de moyenne en grande dimension (c'est-à-dire lorsque la dimension q des entrées $(X_i)_{i\in\{1,\dots,n\}}$ dépasse le nombre d'observations n).

L'objectif de ce projet transversal de mathématiques appliquées et de linguistique est d'apporter un éclairage aux problèmes de simplification automatique de texte, ce qui potentiellement pourrait à termes aider les personnes ayant une déficience auditive à faire face aux difficultés rencontrées. Les travaux sont situés dans des champs variés (statistique, linguistique, informatique) et revêtent un caractère fortement pluridisciplinaire. Cette thèse comporte quatre chapitres que nous allons maintenant présenter.

Plan et contributions de la thèse

Chapitre 1

L'objectif du premier chapitre est d'une part d'introduire les principaux concepts du NLP (traitement automatique du langage naturel) telles que l'analyse syntaxique ou la classification de textes et d'autre part de présenter les formalismes utilisés pour modéliser les problèmes liés à l'analyse de texte (lexicale, syntaxique, sémantique, etc.). Ceci réduit de fait fortement notre champ d'investigation, puisque nous n'aborderons donc que marginalement les questions liées au traitement automatique du langage naturel, et pratiquement pas les question liées à l'adaptation de textes pour les déficients auditifs.

Ce chapitre s'adressant principalement aux lecteurs non familiers avec ce domaine, nous rappellerons les principales définitions et le vocabulaire utilisés en linguistique et en linguistique informatique (appelée

sur les pages "Pass vaccinal" et "Vaccin" de l'espace Covid-19. ANAE est un chatbot qui utilise l'intelligence artificielle pour générer une animation en langue des signes française (LSF) et des sous-titres pour les personnes sourdes ou malentendantes. Cette initiative s'inscrit dans l'effort du gouvernement français dans le cadre de la directive européenne du 17 avril 2019 sur l'inclusion des personnes handicapées et notamment de l'accessibilité aux services.

[5]DALiNK (https://bootcamp.dalink.fr/) construit des stratégies pour développer une communauté, en proposant des services opportuns, en identifiant les centres d'intérêts et en étudiant les nouvelles combinaisons gagnantes pour mieux servir les intérêts de la communauté et réagir efficacement aux nouvelles contraintes ou opportunités de marché.

[6]https://openai.com/product/gpt-4

aussi linguistique computationnelle).

Nous passerons succinctement en revue les méthodes existantes de constitution de corpus. Il s'agit d'un passage obligé pour qui veut développer des modèles d'analyse de texte ou de production et de génération de texte par des modèles prédictifs (de type chatGPT). Nous présenterons d'abord les théories linguistiques disponibles et les outils informatiques qui permettent leur mise en œuvre, ainsi que leur évolution dans le temps. Ensuite, afin de se familiariser avec les concepts du traitement automatique du langage naturel, nous poursuivrons cet exposé par la présentation de quelques méthodes largement utilisées pour l'exploration, l'analyse syntaxique de textes, ainsi que leur automatisation rendue possible grâce aux modèles statistiques et aux méthodes d'apprentissage automatique.

Ce premier chapitre sert également de guide pour l'étude, l'analyse et la synthèse selon des processus de découpage et de transformations classiques. Nous détaillerons notamment les processus de tokenisation (qui permettent de découper le texte en petits morceaux porteurs de sens appelés tokens), puis du POS-tagging (qui permet d'attribuer une classe grammaticale à chaque token) et enfin de chunking et de parsing définis plus tard. Quelques aspects techniques de ces méthodes ainsi que leurs limites sont abordés ici.

Chapitre 2

Nous débutons ce chapitre par l'exposé des principaux modèles prédictifs utilisés pour réaliser automatiquement l'étiquetage morpho-syntaxique, sans entrer de manière approfondie dans l'écriture mathématique des objets associés. L'idée est d'estimer une probabilité conditionnelle d'un token ou d'un tag conditionnellement à un contexte construit à partir de mots ou d'une séquence textuelle (de lettres, de stems, de mots ou de phrases). En particulier, la méthode d'entropie maximale employée par Ratnaparkhi (1996) [166] permet de construire des modèles de classifications simples qui reviennent à estimer des modèles log-linéaires ou des modèles logit.

Nous montrons en quoi ces méthodes peuvent s'interpréter comme des méthodes de vraisemblances empiriques généralisées (construites pour l'entropie relative) sous l'existence d'un grand nombre de contraintes s'interprétant comme des moments. Un des premiers résultats de cette thèse est de proposer des extensions de ces méthodes de vraisemblance empirique généralisée capables de s'appliquer à la majorité des tâches utilisées en TALN et de donner des formules explicites (asymptotiques) pour les probabilités conditionnelles (de tokens ou de tags), y compris lorsque le nombre de contraintes est plus grand que le nombre d'observations.

Pour cela, nous combinerons des méthodes de vraisemblance empirique généralisée pénalisée (Owen (1988) [151] (1990), [150]) avec des techniques d'extraction de "features" (ou caractéristiques), qui permettent de projeter des données textuelles dans des espaces numériques. Nous donnerons ensuite deux applications de ces approches :
1) le POS-tagging (l'étiquetage de parties du discours), qui a été largement étudié ces 20 dernières années (voir Ratnaparkhi et al. (1996) [166], Borthwick et al. (1998) [21], Thorsten Brants (2000) [26], Collins (2002) [47], Guyon et al. (2008) [84], Yogatama (2015) [211]),
2) la classification de texte dans le cadre de la simplification automatique de texte qui fera l'objet de développements dans le chapitre 4.

Étant donné que les contraintes (ici des moyennes de caractéristiques) appartiennent à un espace de grande dimension, nous proposons l'utilisation de méthode de pénalisation basée sur la représentation duale du problème d'origine (selon les principes utilisés par Otsu (2007) [149], Chang et al. (2018) [39] et Shi (2016) [179]). Enfin, la même approche sera utilisée pour construire un classificateur qui prend une phrase en entrée et renvoie une sortie binaire indiquant si la phrase est complexe "0" ou simple "1", ce qui permet de construire un indicateur préalable de complexité.

Chapitre 3

Nous obtenons des inégalités avec des bornes exponentielles pour la statistique de Hotelling T_n^2, qui prennent en compte le phénomène de grande dimension du problème. Nous explorons les propriétés de la fonction de survie de ces statistiques pour des échantillons finis/à horizon fini en dérivant des bornes exponentielles pour les distributions symétriques ainsi que pour les distributions générales sous des hypothèses de moments faibles (nous ne supposons jamais l'existence de moments exponentiels). Pour cela, nous utilisons un estimateur pénalisé de la matrice de covariance et proposons un choix optimal pour la pénalité.

Dans de nombreuses applications telles que le traitement automatique du langage naturel, la dimension du paramètre d'intérêt q est plus grande que la taille de l'échantillon n et croît avec n. Considérons par exemple le problème de l'estimation ou du test d'une moyenne de variables dans $\mathbb{R}^q$, avec $q > n$; dans ce cas, la matrice de covariance empirique n'est pas de plein rang et ne converge pas vers la vraie matrice lorsque n tend vers l'infini (voir Johnstone (2001) [98]). Par conséquent, les tests habituels de Hotelling T_n^2 dans un cadre de grande dimension ne sont plus valables. Il est donc important de construire des estimateurs et des procédures de test qui prennent en compte les aspects de grande dimension du problème (comme cela a été fait par exemple par Ledoit et Wolf (2000, 2022) [115, 116], voir également les références de ce travail). Une proposition pertinente est d'utiliser un estimateur pénalisé, non singulier, de la matrice de covariance au lieu de la matrice de variance-covariance empirique dans les tests. Dans cet esprit, Chen et al. (2011) [43] ont obtenu des tests de Hotelling T_n^2 régularisée asymptotiquement valides pour la moyenne dans le cas gaussien, dans un cadre de grande dimension, lorsque n et $q \equiv q(n)$ tendent vers l'infini à partir d'un certain rang. Li et al. (2020) [122] ont étendu ces résultats à certaines distributions sous-gaussiennes spécifiques. Le but de ce chapitre (qui fait l'objet d'un article soumis à Journal of Multivariate Analysis) est d'approfondir ces propriétés de ces tests pour des échantillons à n fixe. Ainsi nous obtenons des bornes exponentielles pour la statistique de Hotelling T_n^2 correctement régularisée, dans le cas de distributions générales, en supposant l'existence de très peu de moments.

Pour cela, nous dérivons des bornes exponentielles pour la statistique de Hotelling T_n^2 régularisée dans l'esprit de Bertail et al (2008) [14], qui ont obtenu des bornes pour des formes quadratiques auto-normalisées (ou la statistique de Hotelling T_n^2) lorsque $q < n$. Nous montrons que pour les distributions symétriques, seuls l'existence des moments d'ordre 2 est requise et nous ne supposons l'existence de moments d'ordre 8 que pour les distributions générales. Dans la mesure où ces résultats sont originaux et forment le coeur théorique de la thèse, nous en détaillons quelques résultats ci-dessous.

Principaux résultats théoriques du chapitre 3

Soient $Z, Z_1, ..., Z_n$ des vecteurs aléatoires i.i.d. centrés avec une distribution de probabilité P, définis sur un espace de probabilité $(\Omega, A, \mathbb{P})$ avec des valeurs dans $\left(\mathbb{R}^{q(n)}, \mathcal{B}, P\right)$ doté de $\|.\|_2$ la norme L_2. Nous notons $\mathbb{E}$ l'espérance sous P. On pose $Z^{(n)} = (Z_i)_{1 \leq i \leq n}$. Lorsque n et $q(n)$ tendent vers l'infini, on remarque que $(Z^{(n)})_n$ définit en fait un tableau triangulaire de variables aléatoires de dimensions variables. Cependant, comme nous nous intéressons aux propriétés des échantillons à distance finie, nous laisserons tomber la dépendance en n. En particulier, nous utilisons q au lieu de $q(n)$. La matrice de covariance de l'observation est donnée par $S^2 = \mathbb{E}(ZZ')$, où Z' est la transposée de Z et S la racine carrée de S^2. La moyenne empirique de l'échantillon est donnée par $\bar{Z}_n = n^{-1}\sum_{i=1}^n Z_i$ et la matrice de covariance empirique est définie ici par

$$S_n^2 = \frac{1}{n}\sum_{i=1}^{n} Z_i Z_i^{'}.$$

Nous rappelons que le T_n^2 de Hotelling, qui peut être considéré comme une forme quadratique de sommes auto-normalisées, est donné par

$$T_n^2 = n\bar{Z}_n' S_n^{-2} \bar{Z}_n,$$

avec, lorsque $q < n$, $S_n^{-2} = \left(S_n^2\right)^{-1}$. Soient ρ_1 et ρ_2 deux nombres réels strictement positifs. Nous considérons des estimateurs de S^2 définis par $\Sigma_n^2(\rho_1, \rho_2)$, combinaison linéaire de la matrice identité avec la matrice de covariance de l'échantillon

$$\Sigma_n^2(\rho_1, \rho_2) = \rho_1 I_q + \rho_2 S_n^2,$$

avec I_q la matrice identité de taille q.

Dans ce qui suit, nous nous intéressons à la statistique de Hotelling T_n^2 qui utilise une combinaison linéaire de la matrice de covariance empirique de l'échantillon et de l'identité, que nous appelons maintenant la statistique de Hotelling T_n^2 régularisée, définie par

$$T_n^2(\rho_1, \rho_2) = n\bar{Z}_n' \Sigma_n^{-2}(\rho_1, \rho_2)\, \bar{Z}_n$$

généralisant la proposition de Chen et al (2011)[43].

Bornes pour la fonction de survie d'une Hotelling T_n^2 régularisée

Nous obtenons dans un premier temps des inégalités Oracles dans le cas d'une distribution symétrique.

Théorème 1. *Supposons que Z a une distribution symétrique, avec une matrice de variance-covariance finie, nous avons alors, sans aucune hypothèse de moment supplémentaire, pour n'importe quel $n > 1$, pour $t > n$, quelque soit $\rho_1, \rho_2 > 0$,*

$$\begin{aligned}
\mathbb{P}\left(T_n^2\left(\frac{\rho_1}{\rho_2}, 1\right) \geq t\right) &= \mathbb{P}\left(n\bar{Z}_n' \Sigma_n^{-2}(\rho_1, \rho_2)\, \bar{Z}_n \geq \frac{t}{\rho_2}\right) \\
&\leq \frac{2e^3}{9} \bar{F}_n(t) \\
&\leq \frac{2e^3}{9} \exp\left(-\frac{(t-n)^2}{4t}\right),
\end{aligned}$$

où F_n est la fonction de répartition d'une distribution de $\chi^2(n)$.
De plus, pour tout $\rho > 0$, on a

$$\begin{aligned}
\mathbb{P}\left(\frac{T_n^2(\rho, 1) - n}{\sqrt{2n}} \geq t\right) &= \mathbb{P}\left(\frac{n\bar{Z}_n' \Sigma_n^{-2}(\rho, 1)\, \bar{Z}_n - n}{\sqrt{2n}} \geq t\right) \\
&\leq \frac{2e^3}{9} \exp\left(\frac{-t^2}{2\left(1 + \sqrt{2}\frac{t}{\sqrt{n}}\right)}\right).
\end{aligned}$$

Nous pouvons obtenir une meilleure borne pour les statistiques de Hotelling T_n^2 régularisées et pénalisées en nous appuyant sur les résultats de Pinelis (1994) [157] et Laurent et Massart (2000) [110] (voir p.24 de leur article) qui contrôlent la queue de distribution d'une somme pondérée de variables aléatoires indépendantes suivant une loi de $\chi^2(1)$.

Soit $\lambda = (\lambda_j)_{j=1,\ldots,q} \in \mathbb{R}_+^q$ les valeurs propres de S_n^2 (classées par ordre décroissant). Nous définissons pour tout $\rho_1, \rho_2 > 0$, les dimensions effectives suivantes (voir Chen et al. (2011) [43] pour d'autres expressions de ces quantités) :

$$\begin{aligned}
\Theta_1(\lambda, \rho_1, \rho_2) &= \sum_{j=1}^{\inf(n,q)} \frac{\lambda_j}{\rho_1 + \rho_2 \lambda_j} \\
\Theta_2(\lambda, \rho_1, \rho_2) &= \sqrt{\sum_{j=1}^{\inf(n,q)} \frac{\lambda_j^2}{(\rho_1 + \rho_2 \lambda_j)^2}} \\
\Theta_\infty(\lambda, \rho_1, \rho_2) &= \sup_{1 \leq j \leq \inf(n,q)} \left(\frac{\lambda_j}{\rho_1 + \rho_2 \lambda_j}\right).
\end{aligned}$$

Théorème 2. *Supposons que la distribution de Z soit symétrique. Nous avons alors, sans aucune hypothèse de moment, pour tout $n > 1$ et $q > 0$, pour tout $t > 0$ et quelque soit $\rho_1, \rho_2 > 0$,*

$$\mathbb{P}\left(\frac{T_n^2(\rho_1,\rho_2) - \Theta_1(\lambda,\rho_1,\rho_2)}{\sqrt{2\Theta_2(\lambda,\rho_1,\rho_2)^2}} \geq \sqrt{2}\left(\sqrt{t} + \frac{\Theta_\infty(\lambda,\rho_1,\rho_2)}{\Theta_2(\lambda,\rho_1,\rho_2)}t\right)\right) \leq C\exp(-t).$$

avec $C = 3824$. De manière équivalente, nous avons pour la statistique de Hotelling pénalisée, pour $n > 1$ et $q > 0$, pour tout $t > 0$ et quelque soit $\rho > 0$,

$$\mathbb{P}\left(\frac{T_n^2(\rho,1) - \Theta_1(\lambda,\rho,1)}{\Theta_2(\lambda,\rho,1)} \geq \sqrt{2t} + \frac{\Theta_\infty(\lambda,\rho,1)}{\Theta_2(\lambda,\rho,1)}t\right) \leq C\exp\left(-\frac{t}{2}\right).$$

Les bornes du théorème ci-dessus peuvent être utilisées en pratique pour effectuer des tests, en particulier pour la détection d'anomalies dans le cadre de l'apprentissage statistique. Voir par exemple la littérature sur les systèmes de détection d'intrusion utilisant des cartes de contrôle multivariées basées sur T_n^2 de Hotelling (par exemple Tracy et al. (1992) [195] et d'autres travaux de ces auteurs).

Dans un second temps, nous obtenons des inégalités avec paramètres estimés dans le cas de distributions générales sous quelques hypothèses de régularités standards (voir Ledoit et Wolf (2004) [115]). Considérons Λ^2 la matrice diagonale des valeurs propres de S^2 et O la matrice des vecteurs propres associés. Les valeurs propres sont notées $\mu_1, ..., \mu_q$ avec $\mu_1 \leq \mu_2 \leq .... \leq \mu_q$. Nous avons $S^2 = O'\Lambda^2 O$. Maintenant, pour $i = 1, \ldots, n$, nous définissons

$$Y_i = OZ_i \quad \text{avec } Y_i = (Y_{i,1}, ..., Y_{i,q})'.$$

Afin d'obtenir un estimateur adapté aux matrices de variance-covariance en grande dimension, Ledoit et Wolf (2004) [115] ont étudié le minimum de

$$\mathbb{E}\left(\left\|\Sigma_n^2(\rho_1,\rho_2) - S^2\right\|^2\right).$$

Cette minimisation peut être considérée comme un problème de projection dans un espace de Hilbert pour les matrices aléatoires, équipé du produit scalaire $\langle A, B\rangle_{\mathcal{H}} = \mathbb{E}\left[\langle A, B\rangle\right]$ avec comme norme associée $\|.\|_{\mathcal{H}}^2 = \mathbb{E}\|.\|^2$, où $\langle A, B\rangle = Tr\left(AB'\right)/q$ représente un produit scalaire de Frobenius modifié.

Ledoit et Wolf (2004) [115] ont montré que cette minimisation conduit à considérer des valeurs optimales de ρ_1^* et ρ_2^* qui nous permettent de définir une pénalité optimale

$$\rho^* = \rho_1^*/\rho_2^*,$$

que nous proposons d'estimer par un estimateur de type plug-in $\hat{\rho}^*$.

Considérons les hypothèses suivantes :

(A_1) $\exists K_0, K_1 > 0$ tel que pour tout n et pour tout $q \geq n$, $K_0 \leq \frac{q}{n} \leq K_1$.

(A_2) $\exists K_2 > 0$ tel que pour tout n et pour tout $q \geq n$, $\frac{1}{q}\sum_{j=1}^{q}\mathbb{E}\left[Y_{1,j}^8\right] \leq K_2$.

(A_3) $\exists K_3 > 0$ tel que pour tout n et pour tout $q \geq n$, $\frac{1}{K_3} < \mu_1 \leq \mu_q < K_3$.

(A_4) $\exists K_4 > 0$ tel que pour tout n et pour tout $q \geq n$,

$$\nu = \frac{q^2}{n^2} \times \frac{\sum_{(i,j,k,l)\in\mathbf{Q}}\left(Cov\left(Y_{1,i}Y_{1,j}, Y_{1,k}Y_{1,l}\right)\right)^2}{Card\left(\mathbf{Q}\right)} \leq \frac{K_4}{n},$$

où $\mathbf{Q}$ désigne l'ensemble de tous les quadruples composés de quatre entiers distincts compris entre 1 et q.

Théorème 3. *Supposons que les hypothèses* (A_1) *à* (A_4) *sont vérifiées, nous avons alors pour tout* $n > 1$*, pour tout* $q > n$*, pour tout* $t > 2n$ *et pour tout* $\epsilon > 0$ *petit,*

$$\begin{aligned}\mathbb{P}\left(T_n^2(\widehat{\rho}_n^*, 1) \geq t\left(1 + \hat{a}_n^* + 2\epsilon\right)\right) &= \mathbb{P}\left(n\bar{Z}_n'\hat{\Sigma}_n^{*-2}\bar{Z}_n \geq t\left(1 + \hat{a}_n^* + 2\epsilon\right)\right) \\ &\leq \frac{2e^3}{9}\left(\frac{t-n}{2}\right)^{\frac{n}{2}}\frac{e^{-\frac{t-n}{2}}}{\Gamma\left(\frac{n}{2}+1\right)} + \frac{C\left(\epsilon\right)}{n\epsilon},\end{aligned}$$

où $\hat{a}_n^* = 1 + \frac{K_3}{\hat{\rho}_n^*}$*, et* $C\left(.\right)$ *est une fonction réelle strictement positive, indépendante de* n*, définie par*

$$\begin{aligned}C\left(\epsilon\right) &= 4K_1\sqrt{K_2}\left(2 + \frac{1}{q} + K_1\right) + 2K_1 G\left(\sqrt{\frac{\epsilon}{2K_1}}\right) \\ &\quad + \frac{4K_1^2\sigma^4}{\epsilon}G\left(\frac{\epsilon}{2\sigma^2 K_1}\right) + \frac{K_3^2}{\epsilon}G\left(\frac{\epsilon}{K_3}\right).\end{aligned}$$

L'expression explicite de G est donnée par une fonction qui contrôle la proximité entre $1/\rho^*$ et $1/\hat{\rho}_n^*$.

Chapitre 4

Ce chapitre 4 met en œuvre les outils des chapitres 1, 2 et 3, ainsi que leurs extensions aux réseaux de neurones à des données textuelles extraites d'encyclopédie. Ainsi nous avons constitué une base de données et un corpus à partir d'extraction de textes ou d'articles de Wikipédia en deux versions : en anglais standard et en anglais simplifié. Nous décrirons cette étape fondamentale qui s'est avérée très coûteuse en temps.

Plus précisément, nous présentons d'abord brièvement les origines des réseaux de neurones et rappelons les principes des méthodes de Deep Learning (DL) ou apprentissage profond, et les principaux termes utilisés dans ce domaine. Nous décrivons les réseaux de neurones convolutifs et récurrents, ainsi que les LSTM et/ou les réseaux avec des couches encodeur-décodeur qui sont mis en oeuvre sur nos données. Les aspects pratiques (calibration des paramètres du réseau) seront abordés ensuite. Nous résumons enfin les performances d'une série d'architectures d'apprentissage profond, que nous proposons sur notre corpus extrait de Wikipedia. Ceci nous permet de construire une mesure de complexité et un simplificateur de texte automatique (certes beaucoup moins performant que chatGPT mais qui permet de comprendre les mécanismes en oeuvre). Lorsque les données sont contextualisées à un thème donné (par exemple le cinéma ou aux artistes), nous obtenons des taux de performances de nos classifieurs de l'ordre 85%. Par ailleurs, une combinaison de ces différentes architectures nous permet de construire un générateur de texte en version simplifiée qui a été testé pour l'instant uniquement sur un petit corpus mais que nous souhaitons mettre en oeuvre sur un corpus beaucoup plus grand (avec les ressources informatiques adéquates). Enfin, les codes python sont fournis en annexe sous forme de liens vers un répertoire en ligne.

Chapter 1

Automatic text simplification

In this first chapter, we give the main vocabulary and definitions used in linguistics and computational linguistics. It also allows us to review, in a panoramic and non-exhaustive way, the existing methods for natural language processing, for storing dictionaries of words or rules, and for constructing corpora (plural of corpus), since these tasks are indispensable when one wants to develop models for text analysis or text production and generation. We will first explain the linguistic theories that are available and the computational tools that allow their implementation as well as their evolution over time. Then, in order to become familiar with Natural Language Processing concepts (NLP), we will explain some widely used methods for text mining, as well as their automation, which has become possible thanks to statistical models and related machine learning methods. In this chapter, the mathematical modeling of these methods will not be detailed. This will be the subject of Chapter 2 which deals more specifically with some classical mathematical methods used in part-of-speech tagging (or POS tagging which means assigning a grammatical class to each word of a sentence). We expose the main models used to automatically perform this task, and then we will also propose an extension of these methods through their substitution by the generalized empirical likelihood and by explaining that the latter is applicable to the majority of the methods used in NLP.

This part aims at describing some linguistic tools that allow to build an automatic text simplification application. It is mainly intended for mathematicians who may not be familiar with this field. For this, we give a series of definitions used in linguistics and natural language processing, first in a general way, then in a specific way for text simplification. The first section gives an introduction to NLP and linguistics, their history, and their evolution over time. The tools, terms and vocabulary used in this field. But it also gives, in a brief way, the computational and mathematical aspects behind it. The first section also provides a non-exhaustive description of the notions of corpus, and textual data representation, as well as their evolution over time. Then, the second section starts by expressing the problem of reading for deaf people and the studies that have been done on this subject. Then we focus on the interest in simplifying texts and the existing methods of automatic text simplification. This second section presents a few methods of text simplification proposed in the literature. The last section provides the framework and explains how we will use all of these theories and employ these tools to address the initial problem.

1.1 Linguistics and Natural Language Processing

To distinguish human language, we usually talk about *natural languages*, as opposed to *artificial languages* or *formal languages* such as computer programming languages or mathematical logic. Communication can be performed in different ways: by speaking and listening, through facial expressions, making gestures, using hand signals (such as the referee of a soccer match or traffic officer while directing cars), using sign language, or through different forms of text. Natural language processing is a domain or

field that includes all research and development aimed at modeling and reproducing, with the help of machines, the human ability to produce and understand linguistic statements for communication purposes. The objective behind natural language processing is to interact with a machine as if it was a real human being. The use of the term natural refers to human language, as opposed to formal languages such as C+ or Java. cf. Tellier (2010) [189], Véronis (2001) [200] ; Yvon (2010) [213] and kibble (2013) [103]. In this chapter, we will focus only on texts that are written and printed on a flat surface (mainly paper, road, street signs, card maps, etc.) and stored in an electronic device to be read by their designated audience or studied by the intended researchers.

Although computers and cell phones can hear a sentence and transcribe it (like Siri, Alexa, and online voice recognition tools), they are still not able to understand the meaning of the sentence, i.e. to understand the semantics underlying it. By hearing, we mean capturing sounds and transforming them into letters, words, sentences and so on. Talking "naturally" with machines as if they were humans is a dream strongly present in people's imagination, especially in occidental countries. In the 1950s, the founder of computer science Turing predicted that in about 50 years, machines would be able to communicate with humans (which is almost the case today with Chatbots like ChatGPT[1] or BLOOM[2]).

As for most information theory fields, especially in artificial intelligence, we can motivate the study and use of NLP by mentioning some main benefits and reasons: the aim is to model a fascinating and ubiquitous skill that is language, in order to test hypotheses about the text, natural language or more generally communication mechanisms; the need to put into practice linguistic theories allowing the machine to communicate with humans; and finally the need to have applications able to efficiently process the huge amount of information available today in digital format (subtitles, automatic assistants like bots, etc. see Yvon (2010) [213] and Tellier (2010) [189]). Many of these and other motivations are reflected in the history of NLP, which is summarized in the next section. NLP is thus a field of knowledge and sophisticated techniques developed around various problems.

Natural language processing is a transversal field, the concepts and techniques it uses are at the crossroads of multiple disciplinary fields such as, in chronological order, linguistics, logic, theoretical computer science, traditional artificial intelligence (AI), but also, more recently, neuroscience, statistics and, in particular, deep learning. It puts its applications, programs and various computer techniques at the service of natural language in order to manipulate textual data, process them, analyze them, understand their meaning and even produce them.

1.1.1 Historical note

To provide some background, it is useful to recall how things have evolved over time, in linguistics, statistics, and computer science in a parallel way, leading to the emergence of NLP and all these new methods that allow the machine to manipulate and generate texts.

At the end of the 15th century and the beginning of the 16th century (1588-1648), Marin Mersenne tackled the study of phonation, from an articulatory, acoustic, and mechanical point of view. At that time, the pioneers of linguistics are largely represented by the authors of descriptive grammars of a given language, the precursors of computer science are the mathematicians who describe general methods of calculation (as to solve equations for example), and the inventors of mechanical "calculating machines" like Pascal and Leibniz. Later in 1660, Antoine Arnauld and Claude Lancelot published the "Grammaire générale et raisonnée" (known as the "Grammaire de Port-Royal", Arnauld (1756) [5]) which describes the rules of language in terms of universal rational principles. Then, between the 17th and 19th centuries, comparative and historical linguistics was predominant. Languages were compared to each other and intersections and divergences were sought in order to deduce general laws and patterns of evolution.

Then, it is from the year 1916 that things started to move. The notes of the course entitled "Cours de linguistique générale" of the Swiss linguist Ferdinand de Saussure (1857-1913) were published by two of his students Charles Bally and Albert Sechehaye (an afterward publication (1989) [58]). His works are considered as the birth of modern linguistics. Saussure introduces several important distinctions

[1] https://openai.com/product/gpt-4
[2] https://bigscience.huggingface.co/blog/bloom

and concepts, characterizing language as the social construction of a system of signs. He considers that language, as a general faculty to express oneself by means of signs, is distinct from speech.

Saussure's analyses were continued and extended in the 1930s and 1940s with the Prague School or Prague linguistic circle, which promoted "structural linguistics". Moreover, some of its best-known members were responsible for the invention of phonology (the study of elementary sounds called phonemes that play the role of distinctive units in a given language), namely Nicolas and Troubetzkoy and Roman Jakobson (1896-1982). While speaking about phonology, let us point out that we will limit ourselves almost exclusively to the treatment of the language under its written form, the treatment of the speech being still, in spite of convergences more and more marked with the treatment of written texts. Analyzing speech requires a lot of signal theory, which is not the object of this thesis. Afterward, between 1951 and 1954, Harris published his work on structural and distributional linguistics (Harris (1951) [88], Harris (1954) [89]). Then, in 1957 the work of Noam Chomsky marks the history of the syntax of the last 70 years (Chomsky (1957) [45, 44]). We will talk more about the latter in the following, after having described the main reasons for the computer progress and development.

So far, we have only talked about the chronological evolution of linguistics, whereas computer science and mathematics have also evolved in parallel leading to computers and digital textual resources. The English mathematician Alan Turing (1912-1954), proposes in 1936 the device later called "Turing machine" which gives a precise mathematical characterization to the concept of algorithm. This proposal can be considered as the beginning of computer science. In addition, the report written by the mathematician and physicist John Von Neumann (1903-1957) in June 1945 [203] (Reprinted in 1975 [204]), in which he elaborated (within the framework of the EDVAC project) the first description of a computer whose program is stored in its memory, defines the construction plan of computers. Subsequently, in 1950 Turing published another fundamental article in which he described a test to judge the ability of machines to think: this test, since called "Turing test" is based on a dialogue game. (see Turing (1950) [197, 196])

After these two major events mentioned above, Saussure (1916) [204] and Turing (1950) [197, 196], there was a meteoric rise in machine translation (MT) research, which gave rise to enormous expectations by researchers and investors since 1950. Later in 1952, Yehoshua Bar-Hillel (1915-1975) was the first full-time MT researcher at MIT. He organized the first conference on machine translation. Thus, the first experiment in MT from Russian into English took place in 1954, with programs having to use just bilingual dictionaries and a few basic restructuring rules. Although the vocabulary was only using 250 words and the grammar 6 rules, this experiment was the starting point for a lot of work in this field. A few Russian sentences, selected in advance, were automatically translated into English. In the same year, the first issues of the Journal of Mechanical Translation appeared. Therefore, when one refers to NLP, one should be aware that historically the first works in this field were mainly about machine translation, with the development of the first (very rudimentary) machine translator in 1954.

The main works presented in the literature then concern the construction and manipulation of electronic dictionaries, since translation techniques essentially consist of word-to-word translation, with a possible reordering of the words. This naive view of translation has led to several debates and examples where it does not work. The point of these examples is that a lot of contextual knowledge on the one hand, i.e. about the situation described, and encyclopedic knowledge on the other hand, i.e. about the world in general, are needed to find the correct translation of a word (as for instance the famous example of the translation of the word pen in the following two sentences: "the box is in the pen", and "the pen is in the box" where pen means in each sentence respectively the instrument for writing and the small enclosure in which farm animals are kept), see Yvon (2010) [213].

The first book on MT was Mechanical Translation published in 1955 by Booth and Locke [127]. Then, in 1959, the Association for the study and development of automatic translation and applied linguistics named ATALA[3] was created (today the Association for Automatic Language Processing). Bar-Hillel exposed in a report in 1960 the enormous technological and linguistic difficulties of translation (Bar-Hillel (1960) [8]), such as the one cited in the example above on the word "pen". Afterward in

[3] https://www.atala.org/association

1962, the first conference on machine translation was organized at MIT by Bar-Hillel.

However, with the publication of the ALPAC[4] report in 1964 (Automatic Language Processing Advisory Committee, a little summary is given here[5]) which established the failure of research in MT, this led to the end of funding and to the almost total disappearance of research in the field. But, since 1975, MT research has taken off again under the impulse of the European Community with the development of the SYSTRAN system. Since the year 2000, MT has seen a dramatic improvement, with online automatic translators such as Google, Reverso etc. In the 2010s, with the advent of deep neural networks and the ability to store data in the cloud and using powerful graphics cards, one translator, DeepL[6], stood out as the best translator in the world. It is based on sequences of texts that are translated into two versions and stocked in the linguee database linguee[7].

During the 1960s, Natural Language Processing (NLP) emerged from MT as computational linguistics. After that, the following events occurred successively, with the creation of the Association for Computational Linguistics (ACL) in 1962, and then the first biannual international conference of computational linguistics *Coling* in 1965, and finally the creation of the journal Computational Linguistics in 1975 (cljournal website[8]).

Now, let us focus on the eminent linguist Noam Chomsky (born in 1928). Similarly to the distinction between language and speech proposed by Saussure, Noam Chomsky distinguishes linguistic competence (knowledge of a language's rules of operation) from performance (actual implementation of these rules in comprehension or production). He was only interested in competence, arguing that performance is a matter of psychology. According to him, Linguistics must indeed study language independently of its biological support (such as brain areas, genes, etc.), as if we were dealing with an ideal fictitious and abstract speaker. For him, the goal of any linguistic theory is the explanation of the grammatical judgments of which all speakers of a language are capable (especially when it is their mother tongue). He considers that the surface structure (i.e. the syntax) of an utterance or text determines its deep structure through the semantic relations it expresses. In 1957, he published his founding work "Syntactic Structures" (Lees (1957) [118]). The approach was then developed during the 1960s, and extended in particular in the theory of "generative and transformational grammars" which later became the standard theory of syntax as a benchmark.

The mathematical characterization of the notion of "generative grammar" in the 1960s gave rise to what is known as Chomsky's hierarchy. This hierarchy is well known and used by computer scientists because it characterizes language families of increasing complexity. Indeed, this hierarchy represents the foundation of the theory of formal languages, the branch that studies the properties of artificial languages like computer programming languages. Another similar hierarchy will be discussed in the last chapter. Formal semantics then explode during the '70s and '80s, allowing knowledge representation and reasoning formalization, see Minsky (1974) [144], Schank (1972) [173], Schank (1974) [174], Shapiro (1979) [176] and Sowa (1976) [182]. This period also sees the incorporation of pragmatics (the study of the use of language in context, i.e. in concrete situations), in modeling. Expert systems, based on symbolic models of the same kind, are then widely used in AI.

The '90s are marked by the significant increase in the storage and computational capacity of today's computers, as well as the rapid development of the Internet, allowing the emergence of a new form of linguistics, so-called corpus linguistics based on the exploitation of texts in digital format (Stefanowitsch (2020) [184], Perrez (2021) [155]). Tony McEnery (2019) [139] states that "...corpus linguistics is not a branch of linguistics in the same sense as the syntax, semantics, sociolinguistics and so on. All of these disciplines concentrate on describing/explaining some aspect of language use. Corpus linguistics in contrast is a methodology rather than an aspect of language requiring explanation or description. A corpus-based approach can be taken to many aspects of linguistic inquiry...". This more empirical linguistics, based on data rather than on abstract formal models, makes intensive use of numerical and

[4] https://nap.nationalacademies.org/resource/alpac_lm/ARC000005.pdf
[5] https://pangeanic.co.uk/wp-content/uploads/sites/2/2014/04/ALPAC-1996.pdf
[6] https://www.deepl.com/translator
[7] https://linguee.fr
[8] https://cljournal.org

statistical calculations. This evolution is parallel to the progress of empirical statistics, in particular machine learning, a branch of AI dedicated to writing programs that improve with experience, thanks to examples. The idea is to use corpora to automatically learn to calibrate and fix the parameters of models, in order to be able to generalize them and use them on new data. This approach, which is predominant in current research, is discussed in more detail in Chapter 2 and especially in Chapter 4. Note also that distributional analysis has regained some interest in recent years. The diffusion of the Internet and electronic documents has indeed facilitated the constitution of digital corpora, on which some of the tests promoted by this analysis could be programmed. Hence, the number of publications in the field of NLP has been increasing continuously during the last decade. The graph in the figure (1.1) represents the evolution of the annual number of publications on NLP extracted from the following source[9], hopefully containing the maximum amount of information (contained in all journals).

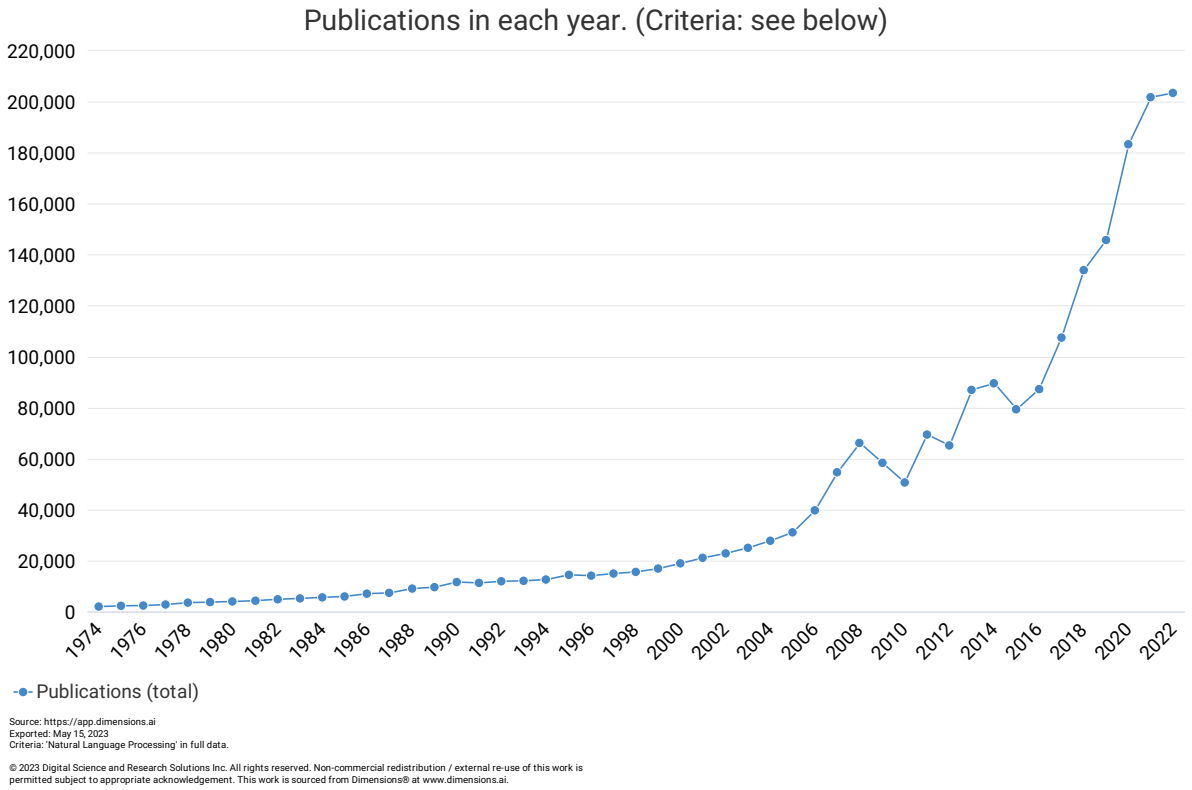


Figure 1.1: Number of publications on NLP between 1972 and 2022.

Recently (during the last ten years), deep neural networks are in full effervescence, especially since the appearance of powerful computing tools such as advanced computers equipped with powerful graphic cards like GPU, virtual machines with large storage and computing capacities in the cloud, which allow to build neural networks with a large number of parameters and layers and train them on a large number of observations, like GANs which allow generating text, LSTMs, RNNs, or more recent network architectures such as Encoder-Decoder, Transformers and Attention mechanisms that are used in BERT (Bidirectional Encoder Representations from Transformers, see Devlin et al. (2018) [59]) and ChatGPT. The last chapter gives more details about these particular settings of neural networks used on textual data.

[9] https://app.dimensions.ai/analytics/publication/overview/timeline?search_mode=content&year_from=1973&year_to=2022&search_text=Natural%20Language%20Processing&search_type=kws&search_field=full_search

1.1.2 Natural Language Processing in practice

Today, anyone who works on texts studies, analyzes, or synthesizes them, starts by pre-processing them according to the following scheme: first, tokenization allows to cut the text into small pieces of meaning called tokens, then POS-tagging allows to assign a grammatical class to each token. Sometimes, we may even need to build the syntactic tree of the studied text called the parser. So, what do tokenization, POS-tagging, chunking and finally parsing mean? This is what we will see in the following. In this section, we will first define these concepts which are part of the main methods used in NLP. We will then explore the technical aspect behind these methods and indicate their limitations.

Tokenization

Tokenization is considered a "low-level" processing, consisting in separating a sentence or a sequence by breaking it up into elementary units carrying meaning (morphemes, words, etc.) called tokens. Here we are considering word tokenization. Even if the task seems to be easier in French and English because of the presence of explicit separators (spaces, punctuation marks, etc.), it is still a relatively complex task in an automatic way, due to the ambiguity of these separators. Moreover, the semantics of these separators may vary from language to language. Here are some typical examples:

, separates propositions, but also the decimal and numerical part of real numbers.

. marks the end of sentences, but also appears in abbreviations like (Mr. Eiichiro) and acronyms (U.S.A.).

The same problem appears for hyphens "-" and apostrophes (see Yvon (2010) [213]). A good Tokenizer (segmenter) must be able to identify all the possible uses of punctuation marks in the written language. In spite of the difficulties that a tokenizer can have to split a text into small pieces, thanks to the regular expressions[10], it is now possible to tokenize any type of text with very high precision, even perfectly for certain languages, in particular English. A second problem, corresponding to a "low level" processing of electronic texts, is related to their format. Whereas only a few years ago, computer documents were stored in a relatively poor format (usually non-emphasized ASCII text), today's documents are available in enriched formats (XML, HTML, RTF, etc.) that contain a wealth of information such as the topic, end-of-paragraph marks, font changes, etc. This information obviously informs about the structure and content of the document, in the same way that the intonation of a sentence informs about its content. This enrichment has led to the production of new annotated corpora that can be used for different tasks.

Tokenization is important for a variety of reasons, including the possibility of studying word repetitions, calculating word frequencies, estimating the richness of the vocabulary, comparing documents, in machine translation, especially for word-to-word translation, etc. So far, we have only presented word tokenization. Tokenization in a general way is the task of cutting a text into small pieces, in a hierarchical way. From text to paragraph, from paragraph to sentence, then from sentence to tokens (or words). Thus, we could also talk about sentence tokenization, which consists in cutting a paragraph into sentences, which is the subject of research working on sentence boundaries. But today, we could say that it is perfectly mastered by NLP and text processing tools, such as the *nltk* library of *python* which provides a "*sent tokenize*" function that allows splitting a paragraph into sentences. This task of sentence tokenization is also known as the sentence boundary detection problem, which received so much attention in the late '90s. Many freely available natural language processing tools require their input to be divided into sentences. Ratnaparkhi (1998) [164] described how to accomplish this using a single model derived from a maximum entropy approach (see also Brill (1994) [28]). Others perform the division implicitly without discussing performance. Here is an example of sentence and word tokenization. Consider the following text fragment,

"He called Mr. Green at 2 p.m. in St. Louis, Mr. White did not answer. He then left him a voice mail message."

[10] a sequence of symbols and characters expressing a string or pattern to be searched for within a longer piece of text.

Ratnaparkhi (1998) [164] describes in his thesis how a computer program can tell which of the .'s, if any, denote actual sentence boundaries, using the maximum entropy approach (see the next chapter for mathematical details). Then, a sent-tokenizer would take this sentence as input and return the following two sentences

"He called Mr. Green at 2 p.m. in St. Louis, Mr. White did not answer."

"He then left him a voice mail message."

And the word tokenization for the second sentence obtained in the sentence tokenization step will return

$$\{\text{"He", "then", "left", "him", "a", "voice", "mail", "message", "."}\}$$

Stemming and Lemmatisation

By combining elementary sounds which, by themselves, do not mean anything, we end up succeeding in "saying" something, that is to say, in referring to something in the world having a meaning. It is a considerable qualitative advance that justifies that one associates a specific level of analysis with it. This new level corresponds to what common sense identifies by the notion of "word". But this notion is not very relevant for linguistics, which prefers to use the "morpheme" term. Thus, the first level that comes in addition to the low level is the "morphological level". Let us recall that morphology covers the studies of lexical forms construction using minimal linguistic units having a form and carrying meaning, called morphemes. There are two main categories of morphemes: radicals (an open category of free morphemes that can be found alone) and affixes (including prefixes, and suffixes that belong to a closed category of linked or connected morphemes). Splitting a word or a token into several morphemes is called stemming.

So "stemming" allows to get stems, by isolating the root of a word and the affixes, which can then play a role in the construction of words, as in inflectional morphology (modification of the canonical form of a word to give it an inflected form expressing its morphological properties in the phrase), derivational morphology (creation of new words from a given radical) or compositional morphology (creation of new words from several radicals).

Lemmatization, on the other hand, allows us to always return to the same reference words (the masculine singular form for nouns, the infinitive for verbs, etc.). It is important to note that, traditionally, morphemes were divided into two categories: lexical morphemes and grammatical morphemes (see Tellier (2010) [189]). But this distinction will not be made here, and we will keep only the first repartition, i.e. affixes and radicals.

Example of stemming and lemmatization Let's consider the following sentence

" *The little mice ate the fresh desserts that my parents had brought from the trip*"

Lemmatizing this sentence leads to

The	little	mice	ate	the	fresh	desserts	my	parents	had bought	from	their	trip
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
The	little	mouse	eat	the	fresh	dessert	my	parent	buy	from	their	trip

This is one approach, other lemmatization approaches can transform *had bought* into the following lemmas *"had"* $\rightarrow$ *have, "bought"* $\rightarrow$ *buy.* Here are another lemmatization examples

$$\text{cats} \rightarrow \text{cat} \quad ; \text{ cacti} \rightarrow \text{cactus} \quad ; \text{ geese} \rightarrow \text{goose} \quad ; \text{ rocks} \rightarrow \text{rock}$$

$$\text{better} \rightarrow \text{good} \quad ; \quad \text{run (as a verb and as a noun), runs, running} \rightarrow \text{run}$$

For stemming, let's consider the following example

"It is very important not to be lazy while you are climbing mountains."

The stem corresponding to each word in the previous sentence is given in the following list

['it', 'is', 'veri', 'import', 'not', 'to', 'be', 'lazi', 'while', 'you', 'are', 'climb', 'mountain', '.']

Part-of-Speech Tagging

In corpus linguistics, part-of-speech tagging (POS tagging or POS tagging), also known as grammatical tagging, is the process of tagging a word in a text (corpus) as corresponding to a particular part of speech, on the basis of its definition and context. In other words, it is the task of assigning to each word in a sentence its grammatical category. These categories are called tags, and the list of these categories is called tagset. Basically, a tagset is a finite list of grammatical categories. There are several tagsets, one of the most commonly used tagsets is the collection called universal tagset, which contains the fewest possible number of tags. It just says if the word is a **pronoun, noun, verb, determinant, adjective, adverb, or punctuation**. So it contains 7 tags in total and it doesn't give any grammatical detail such as word flexions, derivation, verb tenses, etc. It can be divided into two categories:

- open POS-tags (lexical words containing a description of the universe) like nouns, verbs, adjectives and adverbs.
- closed POS-tags (grammatical words that are more like language tools) such as conjunctions, determiners, prepositions and pronouns.

Other tagsets exist, among the most well-known are Penn Treebank[11] (also called wsj tagset by reference to wall street journal) used in Penn Treebank corpus at the University of Pennsylvania. It contains 36 tags, or a total of 46 tags if punctuation tags such as commas, periods, opening and closing brackets, ... are included in the count (Marcus et al. (1993) [131]). The Penn Treebank, in its eight years of operation (1989-1996), produced approximately 7 million words of part-of-speech tagged text, 3 million words of skeletally parsed text, over 2 million words of text parsed for predicate-argument structure, and 1.6 million words of transcribed spoken text annotated for speech disfluencies (see Marcus et al. (1993) [131], Taylor (2003) [188]). There are also other schools that have proposed a richer and more fine-grained tagset. The Brown tagset, a richly grounded dataset inspired by child language acquisition, used a selection of about 80 parts of speech for example (87 tags exactly, see Brown tagset[12]). 61 POS-tag was used in the C5 version of British national corpus (57 plus 4 punctuations). Afterwards 149 tags for the actualized version C7. POS-tagging algorithms aiming to automate this task fall into two distinctive groups: rule-based and stochastic. See figure 1.2.

[11] https://web.archive.org/web/20131109202842/http://www.cis.upenn.edu/~treebank/
[12] https://web.archive.org/web/20080706074336/http://www.scs.leeds.ac.uk/ccalas/tagsets/brown.html

CC	Coordinating conj.	TO	infinitival *to*
CD	Cardinal number	UH	Interjection
DT	Determiner	VB	Verb, base form
EX	Existential there	VBD	Verb, past tense
FW	Foreign word	VBG	Verb, gerund/present pple
IN	Preposition	VBN	Verb, past participle
JJ	Adjective	VBP	Verb, non-3rd ps. sg. present
JJR	Adjective, comparative	VBZ	Verb, 3rd ps. sg. present
JJS	Adjective, superlative	WDT	Wh-determiner
LS	List item marker	WP	Wh-pronoun
MD	Modal	WP$	Possessive *wh*-pronoun
NN	Noun, singular or mass	WRB	Wh-adverb
NNS	Noun, plural	#	Pound sign
NNP	Proper noun, singular	$	Dollar sign
NNPS	Proper noun, plural	.	Sentence-final punctuation
PDT	Predeterminer	,	Comma
POS	Possessive ending	:	Colon, semi-colon
PRP	Personal pronoun	(	Left bracket character
PP$	Possessive pronoun	)	Right bracket character
RB	Adverb	"	Straight double quote
RBR	Adverb, comparative	‘	Left open single quote
RBS	Adverb, superlative	“	Left open double quote
RP	Particle	’	Right close single quote
SYM	Symbol	”	Right close double quote

Figure 1.2: Penn-Treebank tagset. See A. Taylor & al. (2003) [188].

Anna Feldman (2010) [68] mentioned several POS tagging advantages, among which the following three are listed:

- Annotated Corpora with POS tags are very useful in linguistic research for finding frequencies or instances of particular constructions in large corpora (see Meurers (2005) [141]).
- Knowing the part-of-speech information of each word in an input sentence also helps in determining a correct syntactic structure in a given formalism. So POS information can provide a useful basis for syntactic parsing.
- Having a word's POS is useful in morphological generation (mapping a linguistic stem to all matching words for example) since knowing a word's POS gives us information about which morphological affixes it can take. This knowledge is necessary when one aims to extract verbs or other important words from documents, which later can be used for text translation or simplification for example.

Example of POS tagging POS tagging will be discussed in more detail in the next chapter, where methods and mathematical models for automating this task will be discussed. You can find an online tagger here[13]. Here are two examples of POS tagged sentences:

"I saw a girl with a telescope." ; *"The grand jury commented on a number of other topics."*

I	saw	a	girl	with	a	telescope	.
↓	↓	↓	↓	↓	↓	↓	↓
PRP	VBD	DT	NN	IN	DT	NN	.

The	grand	jury	commented	on	a	number	of	other	topics	.
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
DT	JJ	NN	VBD	IN	DT	NN	IN	JJ	NNS	.

[13] http://ucrel-api.lancaster.ac.uk/claws/free.html

Here are some other examples of POS tagging.

Fruit flies like a banana. *Time flies like an arrow.*

Fruit	flies	like	a	banana	.
↓	↓	↓	↓	↓	↓
JJ	NN	VB	DT	NN	.

Time	flies	like	an	arrow	.
↓	↓	↓	↓	↓	↓
NN	VB	PRP	DT	NN	.

In the two sentences "*Fruit flies like a banana.*" and "*Time flies like an arrow.*" notice that the words *flies* and *like* are ambiguous. In the first sentence, *flies* is a noun and *like* is a verb, while in the second sentence, *flies* is a verb and *like* is a preposition. It is this ambiguity that makes this tagging task difficult.

Chunking

Chunking is the task of grouping sentence elements (sequences of words) into sequences called chunks. A chunk is the smallest sequence of words to which a category can be associated, such as "nominal group" or "verbal group". But such a group is only a chunk if it does not contain another group of the same nature. For example, in "the cat of the Neighbor", there are in fact two distinct chunks: "the cat" and "the neighbor". But this unit does not bring enough new properties to justify going to a fundamentally new level of analysis. The only intermediate structure that we will use in the following is the syntagm. A syntagm is a word or a sequence of consecutive words to which one can associate a syntactic category, on the basis of the criterion of substitutability (in a given sentence, one can replace a syntagm with another of the same type or category without altering the syntax of the sentence, the meaning, however, changes unless the use of synonyms). But this notion, as relevant as it may be, does not justify in itself that we devote a specific level of analysis to it, since, precisely, the information that we can associate with it is already present at the level of words (morphological and lexical levels). So we will see that in practice, the term chunk is also used to designate a sequence of chunks. In part of speech tagging, we tag individual words. Chunking works on top of POS tagging and it chunks together a set of tokens like a Verb phrase or a Noun phrase. It is a very important concept if we are working with unstructured data and aiming to obtain information from it.

In the particular task of Noun or Nominal Phrase chunking, it follows rules which determine if the context it takes into consideration represents a Noun phrase, and these rules can be implemented in practice. In classical computer programs such as *Python*, regular expressions are often used to query the machine for particular patterns in textual objects (regexp[14]). The technique was initially used to manipulate strings. But it has been extended to searches on tokens or words, sequences, etc. An example among these rules would be to build a function (via regular expressions) that finds a determiner followed by one or more adjectives and then a noun, the chunk will then be labeled as a nominal phrase. Using the POS-tags defined above we can express this regular expression using the formula $1 \times \text{DT} + x \times \text{JJ} + 1 \times \text{NN} = \text{NP}$ for $x \in \{0, 1, 2, ...\}$ with DT = Determiner, JJ = Adjective, NN = Noun and NP = Noun Phrase. For example "The high white castle" is a noun phrase that matches this regular expression and that corresponds to $x = 2$ and "the rabbit" is another one corresponding to $x = 0$.

Example of Chunking Let us start again from the assignments of the words present in this proposal to these categories: "*Fruit flies like a banana.*" and "*Time flies like an arrow.*".

Fruit	flies	like	a	banana	.		Time	flies	like	an	arrow	.
↓	↓	↓	↓	↓	↓	**and**	↓	↓	↓	↓	↓	↓
JJ	NN	VB	DT	NN	.		NN	VB	PRP	DT	NN	.

[14] https://www.w3schools.com/python/python_regex.asp

First of all, the group "Fruit Flies" can be replaced by a proper noun like "Alfred" or a noun like "Monkeys". We will call the set of all successions of lexical units which can substitute these sequences, the class of "nominal phrases" , for short NP. We have already isolated in our initial statement adjacent groups that can be visualized by labeled parenthesis as follows

$$\text{"(Fruit flies)}_{\mathbf{NP}} \text{ like (a banana)}_{\mathbf{NP}} \text{ ." and "(Time)}_{\mathbf{NP}} \text{ flies like (an arrow)}_{\mathbf{NP}} \text{ ."}$$

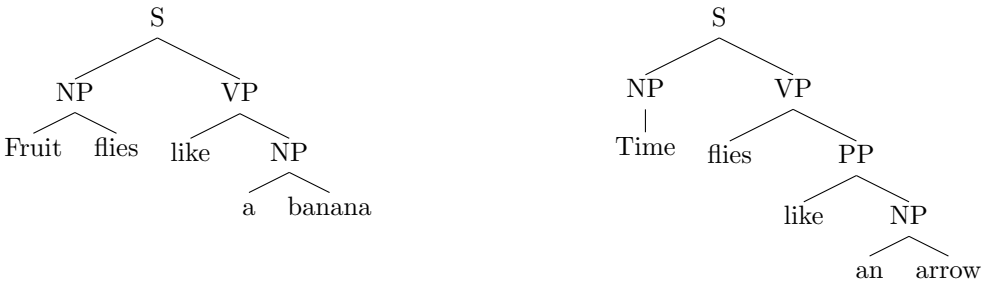
Then, we can realize that it is possible to replace the group "like an arrow" with other noun phrases preceded by a preposition like "as a bolt from the blue" to keep semantics which is not obligatory in this case. Such sequences are called "prepositional phrases", abbreviated as PP. Its identification in our second sentence leads to the following new parenthesis (the label of the group is attached to the closing brackets that delimit it)

$$\text{(Time)}_{\mathbf{NP}} \text{ flies [like (an arrow)}_{\mathbf{NP}}]_{\mathbf{PP}} \text{ .}$$

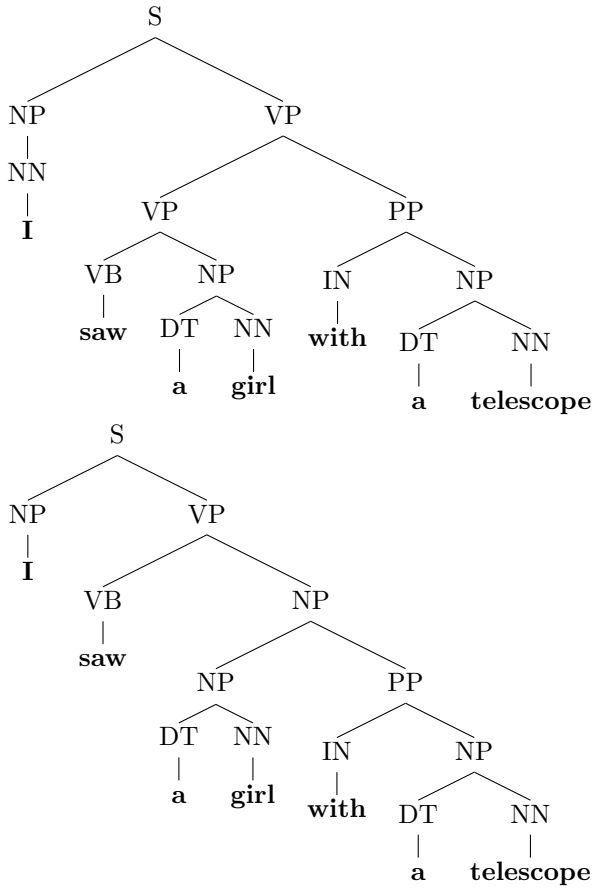
And so on, grouping any meaningful sequence into a group that can be replaced by a similar one, until the parser (the grammatical tree) is obtained. A syntagm is therefore a group of words that corresponds to a sub-tree of a complete syntactic parse tree. For example "like a banana" is a verbal phrase. This explains the term "syntagmatic" which is often associated with Chomskyan grammar. Flana suggests that the English word "sentence" should be translated as "syntagm" rather than "phrase" in French. We can immediately notice one of the biggest problems of chunking, which is the need to assign an order in which to group words into chunks, which could be decisive in some cases where there is ambiguity (see the example 1.1.2 below). That's why most people using chunking methods have to fill in chunking rules, i.e. an order in which to apply chunking, either manually or via methods such as regular expressions. So we could call the method semi-automatic.

Parsing

Parsing is the process of analyzing syntactically a sentence, leading to a tree structure of the sentence, where each sub-tree represents a chunk, and each leaf represents a tag (or word)... A natural language parser takes a sentence as input and determines the labeled syntactic tree structure that corresponds to the interpretation of the sentence. For example, the different part-of-speech assignments for the word flies and likes lead to different parse trees, and different interpretations



This is another example that illustrates the problem of building a tree. Moreover, if it was possible, it would allow us to solve the problem of ambiguity, and we know of course that it is impossible unless we arbitrate between the choices or the author or the one who produces the text is there to inform us more about who had the telescope, the enunciator (author, speaker,..) or the girl. Recall that IN means a preposition, NN a Noun and VB a verb, VP, PP, and NP represent respectively Verbal and Prepositional and Noun phrases, a DT is a determiner.



Therefore, the order of chunking and parsing drastically affects the analysis or understanding of a text, it can even alter the meaning of a sentence, as for these last two parse trees, in the first example, we draw the fact that the person who saws the girl was using the telescope. However, in the second example, that's the seen girl who was carrying the telescope.

We have mentioned the main tools that we will use later, but there are other methods for storing and analyzing or synthesizing texts, notably the finite automaton briefly described below. Finally, let us note that the notion of trees that we have used here constitutes a fundamental "data structure" in computer science: for example, the system of folders (or directories) and files, which manages all the organization of the memory of computers, is of an arborescent nature. In the same way, the HTML language, in which all the pages of the Web are written, is also based on a tree-like description of the contents of a page. Computer scientists are therefore used to manipulating such data. The turn of the mathematicians has arrived, especially the statisticians starting from concrete data to build new adapted models. In addition, in recent years, we have had access to tree-structured corpora, i.e. texts that are parenthetically labeled, making the structure of the sentences they contain explicit. These data, put at the service of computer scientists and linguists who want to exploit them, play an important role in current research.

Actually, some traditional parsing systems do not use such trees at all and instead emphasize the notion of dependencies between lexical units. A dependency is an oriented relation between two words. In this tradition, parsing is thus conceived as a network of dependencies (very similar to the examples in Figure 5.6) and not as a tree. The so-called finite automaton, which was much studied by Chomsky and the pioneers of theoretical computer science, from the 1960s on-wards, and a very large number of their formal properties were then made explicit. A short definition is given below. A finite automaton (or finite state automaton) is a device consisting of 3 elements: - A finite vocabulary (set of words or morphemes) - A finite set of states (with at least one initial state and one final state) - A transition

function (which indicates the set of states that can be reached by starting from a state and using an element of the vocabulary). Nevertheless, we will limit our focus to trees and the corresponding syntactic analyses, based on the notion of *substitutability* between groups of words. Since the tree makes visible the common internal construction of a potentially infinite number of different sentences which makes it expressive.

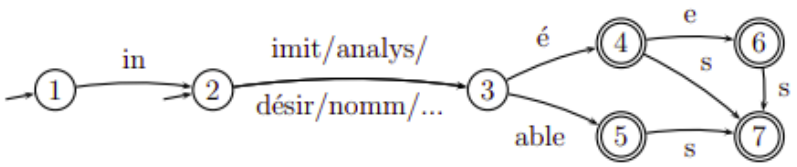


Figure 1.3

After having discussed the levels of textual data analysis, such as the low-level, morphological, lexical, syntactic, semantic, and then pragmatic, we will now discuss the notions of Corpus and data representation. We will not deal with the acoustic aspects of language such as phonetics or phonology mentioned in the NLP history above. We will be particularly interested in the different ways that computer systems can analyze and interpret a text. We will assume that the texts are presented in electronic format. This assumption is reasonable because of the large number of texts available on the Web and all the newspapers and speeches in digital format, which are increasing in number. "This chapter introduces some essential concepts, techniques, and terminology that will be applied in the rest of the course. Some material in this chapter is a little technical but no programming is involved at this stage.

1.1.3 Corpus

Let's first consider a corpus roughly as a large collection of texts (i.e. samples of natural language that are written and printed on a flat surface and stored in an electronic device, or spoken and transcribed from authentic communication situations), and corpus linguistics as any form of linguistic research based on data derived from such a corpus. We will define this concept more precisely in the following paragraphs. Anatol Stefanowitsch (2020) [184] pointed out that any researcher from another field who is not accustomed to the subject and familiar with the predominant theorizing on the subject that has taken place over the past 70 years would wonder why their use should be justified. Indeed, during a few decades, the subject has remained a scientific debate and led to much contention. The main reasons for criticizing this subject (the interest and usefulness of corpus) are that corpora and the data derived from them are necessarily incomplete. They contain only linguistic forms (represented as letters, graphemic strings, and words), but no information about the context, semantics, pragmatics, etc. of these forms; and they do not contain negative evidence, i.e., they only give information about what is possible in a given language, but never or rarely about what is not possible.

The main criticism consists in the lack of information, that is to say, that one will never have all the information and the knowledge of the context surrounding the data that have been stored. Regarding these disadvantages, three major reasons are presented in this paragraph. For a first example, let's consider the example cited above in the historical note on NLP from Chomsky (1964) [46] about the enunciator (the writer or speaker) when communicating information, he represents in his brain the grammar that perfectly reflects the structure of the sentence he wants to express (this process is called the *linguistic competence*), but once he starts to speak or communicate the information, there will be several factors that influence this transmission and that will be added to this linguistic *competence* in order to finally produce as a result a *performance*. He may be confused or have several things in mind, change his plans in midstream, etc. A second example is that corpora, whatever their size, are obviously finite (limited), and thus they can never contain examples of every linguistic phenomenon. For

instance, even the largest currently available linguistic corpora of English or French, such as the BNC[15] "*one-hundred-million-word British National Corpus*" for English or TCOF[16] "*Traitement de Corpus Oraux en Français*" and CRFC[17] "*Le Corpus de référence du français contemporain*" for French, does not contain any instances of some infrequent constructions that are quite present in formal or informal styles. In other words, it seems more likely that these constructions are simply not very common to appear in samples of these million-word corpora. Thus, someone studying the construction might wrongly conclude that it does not exist in British English on the basis of the BNC, TCOF, etc. (see Stefanowitsch (2020) [184]).

For a given distribution of some measurable phenomenon, a representative sample is a subset of the population that is identical to the entire population, whose characteristics perfectly reflect and describe the population as a whole that we can say they are identical with respect to the distribution of the phenomenon under investigation. Similarly, for a given corpus (a sample of texts, a sample of language use), it is representative of a particular language if the distribution of linguistic phenomena (sentences, rhetoric, word ambiguity, words, syntax, etc.) would have to be identical to their distribution in the whole language processed. The way in which researchers and engineers who build corpora try to address this is by including external knowledge such as different manifestations of language and more, leading to what we call sometimes a balanced corpus Stefanowitsch (2020) [184]. Obtaining a representative and balanced corpus remains very difficult when we aim to study a language as a whole. On the other hand, for particular cases such as small specific tasks, the construction of a corpus responding to this purpose seems largely achievable. Nevertheless, if the corpus contains only examples that meet the particular need, there may be the problem of over-representation of a phenomenon seen from a single angle, which can lead to great bias, so their results are unlikely to be generalized even within the limited variety studied.

So even if it seems impossible to build a corpus that is representative of a language in its entirety, there have been several approaches that have aimed at approximating the representativeness of a corpus, by proposing methods and some particular design to make a corpus a little more representative (see Biber (1993) [18]). Among the first linguistic corpora, the first large structured corpus of various genres in our sense is The Brown University Standard Corpus of Present-Day American English (commonly referred to simply by Brown corpus). It is an electronic collection of sample texts of American English, that consists of a little over a million words made up exclusively of running text of edited English prose printed and published in the United States during the year 1961. "So far as it has been possible to determine, the writers were native speakers of American English." (Francis and Kucera (1979) [71]). This corpus first enabled the scientific study of the frequency (see Coltheart et al. (1961) [49]) and distribution of word categories in everyday language use. It is a general language corpus divided into 500 English samples of more than 2000 words each, collected and assembled by Henry Kučera and W. Nelson Francis at Brown University in Rhode Island (Francis (1965) [70]). Even if in the title we find the word "standard", it does not mean that this is presented as *Standard English*, it simply reflects the hope that this corpus could be used for comparative studies where it would be important to use the same data source. Since preparing, entering, and cleaning the data can be a major obstacle in the computer implementation part, the goal behind building this corpus is mainly to provide a structured material of considerable size in a standardized format. Therefore, it clearly does not attempt to be representative of American English in general, but only of a particular kind of written American English in a narrow time span. This is justified if the purpose is precisely to study this specific variety, which is the case for Brown Corpus (Stefanowitsch (2020) [184]).

The term corpus has several slightly or sometimes very different meanings, relative to the need or the academic discipline. In general, it refers to a collection of texts. In literature studies, this collection may be the entire body of work of a particular author (e.g. all the writings of Victor Hugo) or of a particular genre or style, of a particular period (the philosophical works of the first century), or in theology (specific translations of) the Bible or the Thorat. In linguistics, the term corpus is used

[15] http://www.natcorp.ox.ac.uk/
[16] https://tcof.atilf.fr/
[17] https://www.shs-conferences.org/articles/shsconf/pdf/2016/05/shsconf_cmlf2016_11002.pdf

to designate any collection of data (whether texts, sentences, or keywords) constructed for linguistic research purposes, usually in order to answer a particular problem. In the particular field of corpus linguistics, the term corpus refers to a collection of samples of language use, where these instances of language use it contains are authentic, the collection is large and representative of the language or language variety under investigation. Moreover, the texts in such a collection are annotated, not always but more frequently recently, in order to improve their potential for linguistic analysis. For instance, they may contain information about "paralinguistic" aspects of the original data (intonation, font style, speaker humor, etc.), linguistic properties of what is said (parts of speech, syntactic parser), the context and demographic information about the speakers/writers, etc. (see Stefanowitsch (2020) [184] for more details).

As mentioned above, even if building an appropriate corpus is a challenge for researchers, related works have shown the benefit of using statistical tools to build corpora and also to develop language processing techniques. How about the way to construct such a corpus? There is a large literature on this subject, covering all fields of mathematics, linguistics, and computer science. In survey sampling theory, several investigations allow to extract data from a population in order to study them statistically, which would be quite applicable when talking about corpus extraction, by choosing the items to be collected, the context, and the annotations (frequencies, etc.). However, as stated in (Petersen and Ostendorf (2007) [156]), the quality of the system strongly depends on the size of the learning corpus. It is difficult to produce a large training corpus that is of high quality. For example, building a valid morpho-syntactic parser with good performance requires a clean training corpus that contains a very large number of tagged words or parsed sequences (hundreds of thousands to millions). Manual parsing is very tedious and expensive, it can cause several human errors or inconsistencies between the parses produced by different people annotating the same corpus. For some tasks, even with a corpus containing a very large amount of data, one only gets poor results since meaning in language is the product of complex operations and not just a sequence of consecutive words. Moreover, its understanding depends on the context in which the text was written and the reader's general knowledge about the world.

Assuming that the corpus is already collected or available, the elements of statistical sampling theory would also be applicable to the transformation of textual data into numbers and vectors allowing statistical studies on them. In this case, this is where linguistics and computer science would be involved, in the sense that the first one would allow to define and give meaning to the elements that need to be extracted, and the second one would give the material tools that would be used in the practical level by allowing to implement the strategies built-in math and linguistics. These three fields communicate with each other, and it is difficult to dissociate them or to judge through which perspective to approach things and in which order. It is necessary to have a goal first, such as simplifying texts, then to look at the closest field to start with, and finally to keep going back and forth between linguistics, mathematics, and computer science, at each obstacle changing perspective by going back, or by considering a new starting point constantly, each time taking a new path until obtaining satisfactory results. Moreover, this would allow obtaining a path, in order to be able to present a schema to any new person wishing to carry out the same or a similar task, offering him a framework in which he only has to change the corpus with another one having the same design and then to adjust some parameters in order to obtain an automaton of "Text classification" or some specific tasks such as "Automatic text simplification" or more general tasks as for machine translation.

Thus, it is obvious that the quality of the result may be altered or deteriorated depending on the reliability of the corpus used to answer a need. Furthermore, if the corpus is built manually by humans to answer the problem, then the choice of annotations and the type of storage, etc. can be determinant in the relevance and accuracy of the results. This is why we find today that several corpora are built as the project progresses to remain consistent with the objectives and to obtain a robust and reusable corpus, which is still expensive and laborious. But more recently, we notice the emergence of a kind of semi-structured corpus, in large projects mainly led by major web actors such as Google or Wikipedia, allowing access to resources that are produced, corrected, or improved by other users while recording these modifications. We will be covering more about these semi-structured corpora, as well as the structured corpora in the next few paragraphs. This allowed several researchers to put their theory into

practice. For example Yatskar et al. (2010) [210] extracted simple substitutions from the history edits in simple English Wikipedia using probabilistic methods in order to be able to distinguish between edits that remove spam, those that correct spelling errors and those that are actually simplifications. They evaluated their method by selecting 200 editions and then letting three native speakers rank them. They were able to obtain an inter-annotator agreement kappa = 0.69 (Fleiss' Kappa, named after Joseph L. Fleiss is a statistical measure that evaluates the agreement when qualitatively assigning objects within categories for a number of observers). Here is another example of Belder and Moens (2012) [10] working on lexical simplification of texts, who recycled already built resources for a problem that is similar to theirs, lexical substitution. So, the database they used initially contained 430 sentences from Wikipedia with marked words and a list of potential substitutes for each of these words (McCarthy and Navigli (2007) [136], McCarthy and Navigli (2009) [137]). After transforming this dataset by ordering the list of alternatives that can replace the marked word according to their difficulty, Belder and Moens (2012) [10] were able to reuse it for their need, lexical simplification. There are more details on how this database was constructed in their paper, based on annotations from different people. they reconstruct an ordered list of substitute words and finally obtain a lexical simplification tool retrieved from the substitution tool. They also give several metrics that measure the similarity between ordered substitutions (see Ligozat et al. (2013) [124]).

We will distinguish two types of a corpus, a reference corpus called "structured corpus" and the corpus that will be used "semi-structured corpus". In machine learning and Big Data, we distinguish between "structured" data, "semi-structured" data, and "unstructured" data. Nowadays, data are more processed in the context of application development to make information management as simple as possible. The easiest way to manage information, as for the example of relational data, is achieved and made possible using structured data. whereas unstructured data is typically not organized in a predefined clean manner or does not have a predefined data model (in term of computer programming), which make these unstructured data not adapted to a classical relational database. Fortunately for Unstructured data, there are alternative approaches for storing and managing this type of data, among the most known we find the platforms like PDFs, Media logs, etc. Data is becoming an essential component of computer systems and is used by organizations in various business intelligence (BI) and analytical programs. Semi-structured data is information that does not reside in a relational database but that has some organizational properties that make it easier to analyze. With a few pre-processes, you can store them in the relational database (e.g., XML data). In the next paragraphs, we will give the definition of all these notions and explain the difference between them, from a computer point of view first. Then we'll extend these concepts to define and give the difference between structured, semi-structured, and unstructured corpus.

Structured data is the data that conforms to a data model, has a well-defined structure, follows a consistent order, and can be easily accessed and managed by a person or a computer program. Structured data refers to datasets whose elements have a solid and coherent organization and are ready to be used in concrete analysis. These elements have been organized in a formatted repository in well-defined schemes, usually a database, where users can easily search and manipulate the data. In machine learning, it is most often organized in rows and columns with known and predictable content, sometimes even in tensors to extend matrix storage by adding more dimensions to arrays that have only two dimensions (rows and columns). Each column contains a specific type of data, such as dates, text, amounts of money, or percentages. Data that does not match the type of that column is perceived as an error and is rejected. Structured data is well organized which allows some kind of control and auto-correcting since entities in the same group are supposed to have the same attributes for instance. So Data have a well-defined structure that helps in easy storage and access. Residing in fixed fields within a record, data can be clustered manually to group similar entities together and then form relations, categories, or classes and feed some non-supervised learning algorithms. All this, allow data mining to be easy, and therefore knowledge can be easily extracted from structured data. There are several other advantages, here are a few that are far from representing structured data qualities in an exhaustive way: easily scalable in case there is an increment of data (in time or after calculus and processing). operations such as updating, deleting, and substituting are easy due to the well-structured form of data.

BI operations such as Data warehousing can be easily undertaken and finally ensuring the security of data is easy.

Semi-structured data, on the other hand, refers to data that is not captured or formatted in a conventional way. In other words, Semi-structured data does not follow the format of a tabular data model or relational databases because it does not have a fixed or rigid schema. However, it still has some structure. This means that the data is not completely raw or unstructured and contains some structural elements such as tags and organizational metadata that facilitate its analysis. The advantages of semi-structured data are that it is more flexible and easier to scale than structured data. The main characteristics of semi-structured data are the following: First of all, as for structured data, semi-structured data contains tags and elements (Metadata) which are used to group data and describe how the data is stored. Also, similar entities can be grouped together and organized in a hierarchy. Among sources of semi-structured, we can mention e-mails, HTML code and graphs and tables, XML documents and other markup languages, Zipped files, Integration of data from different sources, and web pages. So it provides support to users who can not express their needs directly in SQL since semi-structured data can deal easily with the heterogeneity of sources.

Semi-structured data usually has an irregular and partial structure due to the Lack of a fixed, rigid schema which makes data storage difficult, often found in object-oriented databases in computer science. Some sources have an implicit structure of data, which makes it difficult to interpret the relationship between data as there is no separation of the schema and the data. Moreover, schema and data are usually tightly coupled i.e. they are not only related to each other but linked in the sense of a dependency between them. Thus queries are less efficient as compared to structured data. But despite all these difficulties, semi-structured data have shown their efficiency in many fields and projects, here is an example. In automatic document restructuring in heterogeneous semi-structured corpora, The development of content management systems has profoundly changed the nature of the Web: an increasing number of documents are created automatically and their layout reflects their logical structure. In the same context, querying large databases of semi-structured documents (such as XML) is an open problem that started around the year 2000. Indeed, to query a document whose schema is new, a system must be able to either adapt the query to the document or adapt the document to be able to apply the query to it. Wisniewski et al. (2005) [206] address the problem of transforming semi-structured documents from various sources into a known mediation scheme by proposing a general statistical approach as well as a stochastic approach. They use the INEX corpus to perform a set of experiments to measure the capacity of their model (Wisniewski et al. (2007) [208]). They have also worked on the transformation of heterogeneous HTML documents into XML by applying their model on real corpora (Wisniewski et al. (2006) [207]).

Unstructured Data could be defined simply as follows: if the data is neither structured nor semi-structured, then it is unstructured. The elements of unstructured data may not have the same attributes or properties and the stored collection does not contain sufficient metadata which makes automation and management of data difficult. Dimension, size, and type of the same attributes in a group may differ due to a lack of a well-defined structure, which makes them useless since they can not be processed by computer programs easily, as for censored data.

Structured and semi-structured corpora

The notion of structured, semi-structured, and unstructured corpora are introduced here, and to my knowledge, they have never been introduced before in the same way. Although, a corpus that is meticulously constructed to meet a specific and feasible need (such as calculating word frequencies in the bible) can be considered as a structured corpus. But in general, all corpora are unstructured or semi-structured. They are never absolute because language evolves over time and also from one context to another. Thus, it is obvious that the term structured here is relative, depending on the end user and the application that the corpus will help to construct. As we said earlier, a text corpus is a language resource consisting of a large and structured set of texts that today are mostly stored and processed electronically. Even if the set of texts has some structure, this does not mean that the corpus is structured. Consider the example of a structured corpus subset, if it does not contain all the

information of the corpus it was derived from, then there is a high chance that it will lose its structured setting.

Corpora are used by linguists to perform statistical analysis and hypothesis testing on them, to extract particular occurrences or to validate certain linguistic rules in a given language context. In "search technology", a corpus is a collection of documents (that are going to be searched), and search engines are an application example using "web corpus". A monolingual corpus is a collection of texts in a single language. If the corpus contains texts in several languages, it is called a multilingual corpus. In order to make the corpora more useful for doing linguistic research, corpora are often subjected to a process known as annotation (Developing Linguistic Corpora: a Guide to Good Practice[18] by Geoffrey Leech and Natural Language Annotation for Machine Learning[19] by James Pustejovsky and Amber Stubbs, see also Perrez (2021) [155]). An annotated corpus is often seen as a structured corpus.

POS-tagged or parsed corpus is an example of an annotated corpus. There are several other corpora that use even more advanced structured levels of analysis. In particular, smaller corpora could be completely parsed. These corpora are often called Treebanks or Parsed Corpora which we will briefly describe in the next paragraph since we will use one of them. The challenge of ensuring that the whole corpus is fully and uniformly annotated means that these corpora are generally smaller, containing between one and four million words. Further levels of structured linguistic analysis are possible, including annotations of morphology, semantics, contexts, and pragmatics (such as humor, etc.). Note also that these corpora, although they can be described as structured, are only suitable for the linguistic studies for which they were intended. But they do not allow us to answer all the different problems. For example, a corpus annotated with POS-tags is not better suited for translation than a bi-text built at the sequence level and reciprocally. That is why we prefer to use the term semi-structured in this case. It is because they have a certain structure that can be used to answer other problems, but not the appropriate structure to answer the problem under consideration.

A distributional–relational database, or word-vector database, is a database management system that uses distributional word-vector representations to enrich the semantics of structured data. As distributional word vectors can be built automatically from large-scale corpora (Harris (1954) [89]), this enrichment supports the construction of databases that can embed large-scale commonsense background knowledge into their operations. Distributional–relational models were first formalized by Freitas et al. (2015) [74], Freitas (2015) [73]) as a mechanism to cope with the vocabulary/semantic gap between users and the schema behind the data.

Treebank: In linguistics, a treebank is a parsed text corpus that annotates syntactic or semantic sentence structure. The construction of parsed corpora in the early 1990s revolutionized computational linguistics, which benefited from large-scale empirical data. The term treebank was coined by linguist Geoffrey Leech[20] in the 1980s, by analogy to other repositories such as a seed-bank or blood-bank. This is because both syntactic and semantic structures are commonly represented as a tree structure. Two main groups can be distinguished, treebanks that annotate phrase structure (for example the Penn Treebank) and those that annotate dependency structure (for example the Quranic Arabic Dependency Treebank[21] or the Prague Dependency Treebank[22]).

Parallel text alignment is the process of identifying the corresponding sentences in both sections of the parallel text, allowing for direct comparison and correlation between the sentences in each language. This is particularly useful in various applications, including machine translation. The Loeb Classical Library and the Clay Sanskrit Library are two examples of dual-language series of texts. Reference Bibles may contain the original languages and a translation, or several translations by themselves, for ease of comparison and study; Origen's Hexapla (Greek for "sixfold") placed six versions of the Old Testament side by side. A famous example is the Rosetta Stone, whose discovery allowed the Ancient

[18] https://bond-lab.github.io/Corpus-Linguistics/dlc/chapter2.htm
[19] https://www.oreilly.com/library/view/natural-language-annotation/9781449332693/ch01.html
[20] https://www.research.lancs.ac.uk/portal/en/people/geoffrey-leech(2c241f94-d6a8-4772-8aae-d9bf4952a24d)/publications.html
[21] https://corpus.quran.com/
[22] https://ufal.mff.cuni.cz/pdt2.0/

Egyptian language to begin being deciphered. Large collections of parallel texts are called parallel corpora (see text corpus paragraph). More bibliography and documentation are given in Parallel text processing[23] by Jean Véronis and Marie-Dominique Mahimon. There are several parallel corpora, we can mention Europarl[24], ParaSol[25], EUR-Lex[26] etc. Alignments of parallel corpora at the sentence level are prerequisites for many areas of linguistic research. During translation, sentences can be split, merged, deleted, inserted, or reordered by the translator. This makes alignment a non-trivial task (for more documentation about alignment, see Building and Using Parallel Texts: Data Driven Machine Translation and Beyond 2003[27] and 2005[28]).

Parallel texts find utility in language education as well. Categorically, parallel corpora can be classified into four primary types: (1) A parallel corpus consists of translations of the same document in two or more languages, with alignment at the sentence level or higher. However, these types of corpora (that possess high comparability) tend to be less common than corpora with lower comparability. [218] (2) A noisy parallel corpus comprises bilingual sentences that lack perfect alignment or exhibit poor-quality translations. Nevertheless, the majority of its content consists of bilingual translations related to a specific document. [209] (3) A comparable corpus is built from non-sentence-aligned and untranslated bilingual documents, but the documents are topic-aligned. (4) A quasi-comparable corpus consists of highly diverse and non-parallel bilingual documents, which may or may not be aligned based on their topics. In the field of translation studies, a bitext refers to a combined document containing both the source-language and target-language versions of a given text. (B. Harris (1988) [86]). Bitexts can be generated using an alignment tool or a specialized bitext tool, like a software specifically designed for this purpose, such as GIZA++[29], WAT[30] or vecalign[31], which is used to automatically align the original and translated versions of the same text, typically matching them sentence by sentence. A collection of these aligned texts is referred to as a bitext database or bilingual corpus, which can be accessed and consulted using a dedicated search tool.

1.1.4 Representations

The definition of textual data representation depends on the object to represent and in particular on the considered granularity: a word, a sequence of words, a sentence, a paragraph, a text, or a whole document. Bag of Words (BoW) is an example of a document representation, in the form of a vector where we simply count the frequencies of the words that the document contains (Petersen (2007) [156]). Nowadays, we hear everywhere that Google is becoming capable of understanding what we write to it, that is to say, that it almost understands the sense of words. It is even able to make "calculations" with these words. Quite surprisingly, it was found (Mikolov et al. (2013) [142]) that the similarity of word representations goes beyond simple syntactic regularities. With a word shift technique where simple algebraic operations are performed on the word vectors, it was shown for example that $vector(\text{"King"}) - vector(\text{"Man"}) + vector(\text{"Woman"})$ results in a vector that is closest to the vector representation of the word Queen (Mikolov et al. (2013) [143]). For instance, we can ask the machine to find all the words that are close $(+)$ to the words "King" and "Woman" but far $(-)$ from the word "Man". We get the word "Queen". The same thing if we explore capitals by looking for $vector(\text{"Paris"}) + vector(\text{"Germany"}) - vector(\text{"France"})$ we get vector("Berlin"). It's true that this may sound a bit strange. How is it possible to do mathematics with words? Actually, what we're talking about here is a technology that was developed by Thomas Mikolov (Researcher at Google) in 2013 (Mikolov et al. (2013) [142][143]). It's called Word2vec, and it belongs to the Word Embedding

[23] https://web.archive.org/web/20040417031546/http://www.up.univ-mrs.fr/~veronis/biblios/ptp.htm
[24] https://www.statmt.org/europarl/
[25] http://parasolcorpus.org/
[26] https://www.sketchengine.eu/eurlex-corpus/
[27] https://web.archive.org/web/20060913013656/https://www.cs.unt.edu/~rada/wpt/
[28] https://web.archive.org/web/20060910135811/http://www.statmt.org/wpt05/‘
[29] https://www-i6.informatik.rwth-aachen.de/web/Tools/GIZA++.html
[30] http://phraseotext.univ-grenoble-alpes.fr/webAlignToolkit/
[31] https://github.com/thompsonb/vecalign

algorithms. Basically, it allows a machine to understand what words "mean". We'll come back later on the Word2vec method to understand how it works and give more details about it, but before that, we'll talk more generally about Word Embedding.

Word Embedding was motivated by the aim of improving information retrieval, particularly in search engines like Google and Altavista. For example, when we make a query on a search engine to find a restaurant or a place with mood-enhancing food, let's imagine that this search engine uses Word Embedding. So we could write the word "Sandwich" in the search bar, and then the search engine will expand the search by adding other words that are "similar" (i.e. having the same meaning or close to the word we entered), here, for example, it will be "Hot-dog", "Hamburger", "Burgers", etc. With that, the person making the query is going to find content close to what they're looking for, and that's even if they didn't really say it explicitly, by searching for sandwiches, they get burgers, and that's convenient in a lot of cases. So this is quite useful because it addresses the problem of the void in information search (i.e. when a person does a search but gets no results). To summarize, Word Embedding improves information retrieval, and this is true both for search engines (Google, Yahoo, YouTube..) and conversation with machines (e.g. speech recognition virtual machines such as Alexa and Siri), but it is also indispensable when one wants to perform statistics on texts. Here I am referring to language models, text processing, and machine translation. Obviously, Word Embedding was not invented in a day, and we will go back over the major steps that made this technology possible.

It started in the middle of the 1950s with the brilliant idea of some linguists. We can choose as a starting point 1954 when Zellig Harris published "Distributional Structure" (Harris (1954) [89]), in which he proposed the following hypothesis "Differences in meaning are correlated with differences in distribution". This was then reformulated a few years later by John R. Firth (1957) [69] who said "You can only know a word by its associations", and this is what is called the distributional hypothesis, it's a hypothesis that suggests that words that are found in the same linguistic context have a close or similar meaning. In a simplified way, the linguistic context refers to a window containing the words that are found before and after the word in question.

c=0 The quick brown fox jumps over the lazy dog.
c=1 The quick brown fox jumps over the lazy dog.
c=2 The quick brown fox jumps over the lazy dog.
w represents the Center Word and w represents Context Word.

In the example above, we have the observed word in red (called Center Word), here it is "fox". And in blue, we have the context of the word fox, we call it Context-Words. And for the size, we can choose to take one (c=1), two or three words or more. The idea is that if the word "dog" ("wolf" or "cat"..) is statistically surrounded by the same words as the word "fox" ("*the little ... scratches the tree*" example of a context where we can fill in the blank with one of these words fox/wolf/dog/cat, etc. and the meaning of the sentence remains correct), then it signifies that the meaning of the word dog is close to the meaning of the word fox.

However, this distributional hypothesis has not been directly linked to artificial intelligence, because when this hypothesis was proposed, it was the 1950s and computer science was only in its infancy at the time. Linguistics and computer science were then limited to the application of rules engines, where little statistics and even less machine learning were used. An important step was made by Gerard Salton in 1983, in his publication "Introduction to Modern Information Retrieval", in which he introduced the concept of vector model (Salton (1983) [172]). Basically, the idea is to represent a text as a vector, i.e. a list of numerical values. The simplest way to start building such a vector is to count the words that the text contains. Vectorizing texts this way led to one of the first (and most naive) document representation models - the bag of words. Consider the four following sentences $s_1, ..., s_4$

$$s_1 \longrightarrow \text{"The cat likes its kibble"}$$

$$s_2 \longrightarrow \text{"The dog eats its kibble"}$$

$$s_3 \longrightarrow \text{"The dog barks and drinks"}$$

$$s_4 \longrightarrow \text{"the cat purrs and eats"}$$

Then, by transforming these four sentences into a vectors (via the *Countvectorizer* function of the sklearn library on *Python* for example) we get

	and	barks	cat	dog	drinks	eat	its	kibble	likes	purrs	the
$s_1 \longrightarrow$	(0	0	1	0	0	0	1	1	1	0	1)
$s_2 \longrightarrow$	(0	0	0	1	0	1	1	1	0	0	1)
$s_3 \longrightarrow$	(1	1	0	1	1	0	0	0	0	0	1)
$s_4 \longrightarrow$	(1	0	1	0	0	1	0	0	0	1	1)

The problem with this way of creating a vector by counting words is that the words all have the same weight. for example in the third and fourth sentences, we can see that the word "the" has as much weight as the words "barks" and "purrs", and yet intuitively, we know that the words "purrs" and "barks" contain more semantic information than the determiner "the". Thus, to fix this problem, Salton (1983) [172] proposes to use another numerical value than the number of words. This is called TF-IDF, which stands for "Term Frequency - Inverse Document Frequency".

The Term-Frequency represents the frequency of words in a specific document, for example, the frequency of the words "barks" and "purrs" is 20% in the third and last sentence respectively. The Document Frequency represents the frequency of the documents that contain this central word, and as here there are four documents, only one of which contains the word "barks", so its Document-Frequency is 1/4 Basically, the TF-IDF makes a trade-off between these two measures. A word that appears in very few documents, but very present in a specific document, will have a higher score. While a word that appears in all documents will have a lower score. More precisely the TF-IDF index for a given term t for a document d in a class of document D is computed as follows

$$TF\text{-}IDF\,(t,d,D) = TF\,(t,d) \times IDF\,(t,D)$$

$$\text{with} \quad TF\,(t,d) = 0.5 + 0.5\frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}}$$

$$\text{and} \quad IDF\,(t,D) = \log\frac{N}{|\{d \in D : t \in d\}|}$$

where N is the number of all documents and $f_{t,d}$ is the frequency of the term t in document d. The result for the same example are given below

	and	barks	cat	dog	drinks	eat	its	kibble	likes	purrs	the
$s_1 \longrightarrow$	(0	0	0.445	0	0	0	0.445	0.445	0.565	0	0.295)
$s_2 \longrightarrow$	(0	0	0	0.475	0	0.475	0.475	0.475	0	0	0.314)
$s_3 \longrightarrow$	(0.421	0.533	0	0.421	0.533	0	0	0	0	0	0.278)
$s_4 \longrightarrow$	(0.445	0	0.445	0	0	0.445	0	0	0	0.565	0.295)

By comparing the vectors proposed by TF-IDF with those obtained by the function Countvectorizer in Python, we notice that the word "barks" now has a higher score than the determiner "the". Since it appears in fewer documents, its document frequency is lower, so its inverse makes the score higher. Thus, it has potentially more semantic value.

Once the sentences are transformed into vectors, we can consider them as points in an 11-dimensional vector space. Thus, we can measure the distance between these points (vectors), which would give the distance between the sentences. The distance, or rather the similarity between sentences is usually measured by the cosine of the angle between these vectors. The cosine distance between sentence 1 and sentence 2 is about 50% (0.4848 exactly) in this 11-dimensional vector space. On the other hand sentence

1 is further away from sentence 3, and we can check it by measuring the cosine distance between these two sentences, and we obtain 0.9179 which is almost the double. To summarize: meaning or semantic = context, a context can be transformed into a vector, we can measure the distance between two vectors. So we can measure the semantic distance between two words.

Word2Vec

Let's talk more about the representation of words, and describe in more detail how to embed words instead of sentences. Since early years, many NLP systems are using atomic word representations, also known in the computer sciences field by one-hot encoding. The technique consists of operating in a V-dimensional space where V is equal to the vocabulary size (i.e. the number of words in the vocabulary) and assigning a position to each of the words, putting the value of 1 in the corresponding component and 0 in the rest as follows

$$\text{word}_i = (0, 0, \ldots, 1, 0, 0, \ldots, 0) \text{ with 1 at the } i\text{-th position},$$

and word_i being the i-th word in the dictionary, "apple" for example.

This kind of representation creates a serious problem. For example, if you see in the training data the words "fox" and "quick" co-occurring together, then by only using this kind of representation, there is no possibility for the machine to figure out whether between "dog" and "banana", which one fits that blank spot in the following example "*I made a .... juice*". So, it is obvious that this kind of representations does'nt really fit how human brains perceive these concepts, if we seek to imitate the human brain by the machine in artificial intelligence. When a human being sees the word "food", only certain region of his brain and some neurons are excited and some others are inhibited. When the word is changed from "food" to "breakfast" or "restaurant", you can imagine really few neurons change, because they are quite similar concepts that are close in some sense. But if you change that word "food" to "train", then significantly different set of neurons are excited. So this is representing word as continuous levels of activation, and this is called distributed representation. Neural embedding models aim at recreating this type of representations in a computational way.

This technique was introduced by Mikolov et al. (2013) [142] who have extended previous works using neural network-based language models that have outperformed the classical models used in the literature such as n-grams (Brants (2007) [27]), notably by taking inspiration from the work of Bengio et al. (2000) [11]. Earlier, we used N-gram techniques (mostly with $N = 3$) for language models, but since the progress of machine learning techniques in recent years, we realized that using neural networks approach, for word representations or probability function estimation, improves very significantly on a state-of-the-art trigram model. Since the goal of statistical language modeling is to learn the joint probability function of sequences of words, Bengio et al. (2000) [11] consider that a statistical model of language can be represented by the conditional probability of the next word given all the previous ones in the sequence. In their approach, the model learns simultaneously (1) a distributed representation for each word (a similarity between words) along with (2) the probability function for word sequences, expressed with these representations. That's why we talk about distributional approach.

So Mikolov proposed two new model architectures to obtain word representations that are continuous, from large datasets (Mikolov et al. (2013) [142]). This technique allowed them to outperform all existing methods on multiple degrees of similarity and several tasks, with a lower cost in terms of numerical computation. In their example, it took less than a day to learn high-quality vector representations from a 1.6 billion word dataset (Mikolov et al. (2013) [142]). As a result, they were able to obtain vector-space word representations that are implicitly learned by the input-layer weights, called "continuous space". They found that these representations are curiously good at capturing syntactic and semantic regularities in language and that each relation can be measured or characterized by a relation-specific vector, these vectors are in fact the distances between words with the same regularity, the passage from singular to plural, for example, is represented by almost the same distance between the vectors of the words in singular and their corresponding in plural $vector(\text{"cat"}) - vector(\text{"cats"}) \approx vector(\text{"dog"}) - vector(\text{"dogs"})$ (Mikolov et al. (2013) [143]).

Let us see how it works by using the precedent example "*The quick brown fox jumps over the lazy dog.*". So we have a window that moves across couples of text. The input data is the center word (the word in red below) and the output data (words in blue) is the context word in Skip Gram models, and the opposite in the Cbow models where the center word becomes the output and the context word the input.

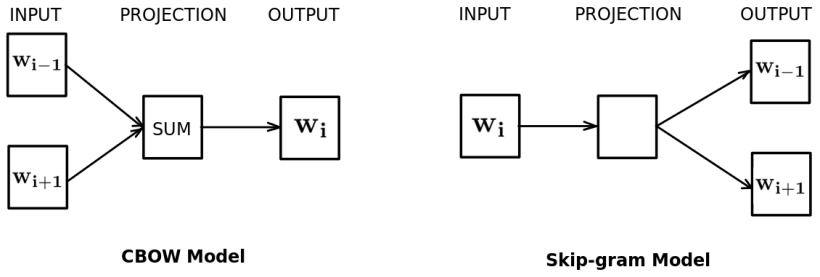


Figure 1.4: CBOW and Skip-gram models (Mikolov et al. (2013) [142])

We can specify the size of the window, but for this example 1 we took a context as a window of one word each side.

quick The quick brown fox jumps over the lazy dog.
brown The quick brown fox jumps over the lazy dog.
w represents the Center Word and w represents Context Word.

Until we end-up with a training pairs with the center word in a side and the context word in the other side, with a blank (or some keyword used to program some variables in data processing tools) either before the starting or after the ending word. Then, the training dataset for a CBOW model with a window of size two is the following

$$\Big\{(x_i, y_i)\Big\}_{1\leq i\leq n} = \Big\{\Big(\text{The brown}, \textbf{quick}\Big), \ldots, \Big(\text{fox over}, \textbf{jumps}\Big), \ldots\Big\}$$

A single-layer neural network is trained using this data. Of course, this data is pre-transformed before using one-hot encoding as mentioned above. Here is what the architecture of the model looks like.

Architecture

Let's consider the CBOW model which consists in taking a central word and predicting the next word which is more convenient to explain. In this example 2 we consider

$$\Big\{(x_i, y_i)\Big\}_{1\leq i\leq n} = \Big\{\Big(\text{The}, \textbf{quick}\Big), \ldots, \Big(\text{fox}, \textbf{jumps}\Big), \ldots\Big\}$$

This means that the window around the word contains only one word on the right. Note that the size of the dataset (the number of words it contains) n is normally different from the size of the vocabulary V, being much larger. Indeed, in the vocabulary, each word is unique, whereas, in the learning dataset, the words are repeated several times with different functions and roles depending on the context. This means that in the following example, words $w_{end} \neq w_V$ since the last word in the learning dataset w_{end} could be any word $(w_i)_{i=1..V}$ in the vocabulary.

Recall that we have $n \gg V \gg d$, corresponding respectively to sample size, vocabulary size and continuous representation vectors dimension. In this simplified example we suppose that the vocabulary is only based on the observed words (listed as they arrived). Then we can code the words as vectors

$$n \left\uparrow \begin{bmatrix} The \\ quick \\ brown \\ fox \\ jumps \\ over \\ the \\ lazy \\ dog \\ \vdots \\ w_{end} \end{bmatrix} \right. \longrightarrow The = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix}, \ldots, jumps = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix}, \ldots, the = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix}, \ldots, w_V = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \left. \uparrow \right\downarrow V$$

Then, when you feed for example "jumps" as input x_i, the network is going to work from left to right with W_1 the $V \times d$ matrix that contains the weights. They are multiplied by x_i before the first layer, and W_2 the $d \times V$ matrix multiplies the output of the hidden layer $h_i = W_1^T x_i$, with d representing the number of neurons. Define $\arg-\max(z_1, ..., z_V) = (0, \ldots, 0, 1, 0, \ldots, 0)$ which indicates the place where the component z_j is the greatest. Then we apply the softmax activation function $\hat{y}_i = f\left(W_2^T h_i\right)$, with typically $f(z) = \arg-\max\left((e^{z_j} / \sum_{k=1}^{V} e^{z_k})_{1 \leq i \leq V}\right)$. For instance we can compute for a given values of W_1 and W_2 the quantities

$$\overbrace{\text{jumps}}^{x_5} = \left. V \uparrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \right\downarrow \Longrightarrow \overbrace{W_1^T x_5}^{h_5} = \left. d \uparrow \begin{bmatrix} 2.1 \\ -1.3 \\ 0.4 \\ 0.6 \\ 0.67 \\ -0.7 \\ 3.25 \end{bmatrix} \right\downarrow \Longrightarrow f\left(W_2^T h_5\right) = \left. V \uparrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \right\downarrow = \overbrace{\text{over}}^{\hat{y}_5}.$$

The true value of this example is $y_5 = x_6$="over". So if $\hat{y}_5 = x_6$ then the result is correct and the error is equal to 0 for this observation in this case, otherwise the error is equal to 1. This is done for the whole database, comparing $\hat{y}_i$ to x_{i+1} for $i = 1 \ldots n$. Now, the model will learn itself. This will be accomplished using a back-propagation and gradient descent algorithms to train the parameters of the model and these weights in W_1 will represent the vector representation of the word. More precisely, for each word$_i$, its vector representation is $h_i = W_1^T x_i$ and since x_i is one-hot encoded, there is only one column of W_1^T i.e one row of W that is selected.

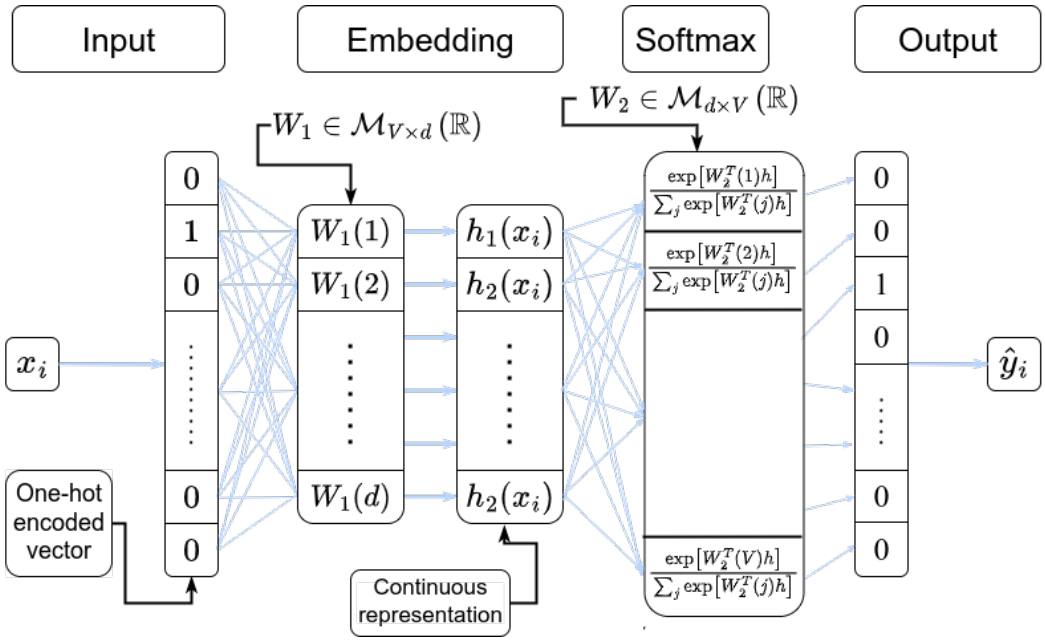


Figure 1.5: Word2vec - CBOW architecture

In this example 2, we only predict the next word. If we want to do it for a window that contains two words as in the example 1, then the only difference is going to be in taking the average instead of taking a single word in the calculation done within the hidden layer as follows, for a context word that contains two words

$$h_i = \frac{1}{2} W_1^T \left(x_{i-1} + x_{i+1}\right) \text{ for the context of two words.}$$

If the context contains 2C words for example, then the formula will be

$$h_i = \frac{1}{C} W_1^T \left(x_{i-C} + x_{i-C+1} + ... + x_{i+C-1} + x_{i+C}\right).$$

Recall that h_i is the i-th word's continuous representation, a vector with values of weights contained in i-th row of weights matrix which is more convenient than the one-hot encoded vectors that are high-dimensional with empty components. Now let's see the performance of the resulted vector word, or embedded words and how they are able to capture semantic information. An interactive training program of the CBOW neural network can be found here[32].

By doing a Principal Component Analysis (PCA), we were able to represent the vectors of words that contain countries and their capital, and it turned out that each country is about the same distance away from its respective capital. The figure 1.6 shows the graphs obtained by representing the first and second principal axes in the graph on the left, and the first and third principal axes in the graph on the right.

[32] https://ronxin.github.io/wevi/

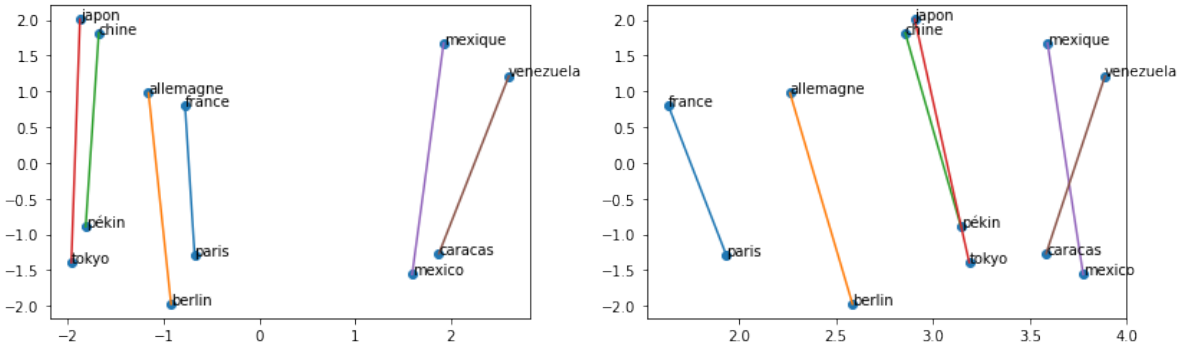


Figure 1.6: (1st, 2nd) axes & (1st, 3rd) axes principal components representation of word vectors.

The distance between each country and its capital is determined (i.e. the distance between the vectors of the words in term of their word2vec continuous representation), and then we calculate the average of all these distances (or the median). We can then build a function that receives a country and adds to it the value obtained (average or median) and returns the word whose vector is closest to it (in terms of L1 or L2 or cosine distance). Here are some results obtained using cosine similarity and the mediane of distances between countries and their capitals. You can find the program code in python here[33], the function *find_capital*.

"uruguay" → montevideo ; **"canada"** → ottawa ; **"équateur"** → quito

We can do this to try getting different genders, singular to plural, and other stuff. For example, if we consider analogies of the following form: A is to B as C is to D. In terms of the word vectors, we should have: $u_a - u_b = u_c - u_d$ where u_w is a word w's vector. Using this, we've been able to create a function that does this analogies. Then with the input , we got

homme - femme + roi = **reine** ⟷ man - woman + king = **queen**

voiture - voitures + camion = **camions** ⟷ car - cars + truck = **trucks**

grand + haut - petit = **bas** ⟷ big + high - small = **low**

ciel - bleu + feu = **rouge** ⟷ sky - blue + fire = **red**

chiot - chien + chaton = **chat** ⟷ puppy - dog + kitten = **cat**

or even something like "positif + négatif - heureux = **malheureux**" which leads after translation to "positive + negative - happy = **unhappy**".

There are several other extensions or variants of word2vec such as BERT (Bidirectional Encoder Representations from Transformers) where the network is deeper with more layers and neurons for example Devlin et al. (2018) [59]. We won't explain how it works, but keep in mind that it is simply a variant of word2vec, in which the architecture of the neural network has been transformed so that it captures more information and details Tenney et al. (2019) [190], but the principle remains the same, the words that are stored in a vector in one-hot encoding (each word is represented by the i-th vector of the canonical basis of $\mathbb{R}^q$ with i the position of the word in the dictionary and q the size of the dictionary) are multiplied by a matrix of weights at the first layer, which selects the i-th row or column of the matrix of weights, and this last one represents the so-called continuous or dense distributed representation of the word.

[33] https://issouani.perso.math.cnrs.fr/

Doc2Vec

Let's focus now on sentence representation instead of word representation. There are two possibilities, either we use again word2vec to build on top a model for the sentences instead of the words based on word vectors, or we build directly a model or a representation on the sentences (Le and Mikolov (2014) [111], Dai et al. (2015) [56], Mueller and Thyagarajan (2016) [145]). Empirically, each approach has advantages and shortcomings. Choosing one depends on the task we want to perform with the resulting vectors. The composition of word vectors in order to obtain higher-level representations for sentences (and further for paragraphs and documents) is a really active research topic (Socher et al. (2013) [181], Kenter et al. (2016) [101], Le and Mikolov (2014) [111], AM Dai et al. (2015) [56], Mueller and Thyagarajan (2016) [145], Lau and Baldwin (2016) [109], Lan and Xu (2018) [108]). There is not one best solution to do this, it really depends on the task we want to apply these vectors on. Some researchers try concatenation, simple summation, pointwise multiplication, convolution, etc. There are several publications on the subject, but ultimately one just needs to experiment and see what fits his problem the best, since there is no theoretical justification but only an empirical one.

Consider a new person working on NLP and trying to perform sentence or document embedding for a classification or a clustering problem, using word2vec which is based on single words to build sentence vectors. Assuming that the word2vec model has already been created using Python's gensim library for example (to reuse pre-trained word vectors) and that the person is wondering to construct a sentence embedding using those vectors. Let's say that the sentence is "the white house", so the vocabulary is composed of the three words "the", "white" and "house" that we suppose mapped as follows

$$\begin{array}{rccc} \text{the:} & [0.00045, & -0.04293, & 0.03045], \\ \text{white:} & [-0.943, & 0.05311, & 0.5839], \\ \text{house:} & [0.442, & 0.0223, & -0.01532]. \end{array}$$

How can we embed this sentence? How can we get the vector of the entire sentence as a whole? The usual approach is to average the vectors of all words in the sentence (dividing the sums by the sentence length). This average vector will represent the sentence vector (Socher et al. (2013) [181])

$$sentence: \quad [-0,16685, \quad 0,010827, \quad 0,19967].$$

Another method could be averaging of Word2Vec vectors weighted with TF-IDF: this is one of the best approaches recommended. It consists of multiplying the word vectors with their TF-IDF scores, then taking the average to represent the sentence vector. There are other different methods to get the sentence vectors, using Doc2Vec for example by training the dataset using Doc2Vec and then using the sentence vectors, instead of a CBOW neural network that guesses a single word using its context, the model will predict a center sequence or a center sentence using the context of this one.

Doc2vec was proposed by Le and Mikolov (2014) as an extension to word2vec (Mikolov et al. 2013 [142]) to learn document-level embeddings. According to Kenter et al. 2016 [101], "simply averaging word embeddings of all words in a text has proven to be a strong baseline or feature across a multitude of tasks", such as short text similarity tasks. A variant would be to weight word vectors with their TF-IDF to decrease the influence of the most common words. A more sophisticated approach developed by Socher et al. (2013) [181] is to combine word vectors in an order given by a parse tree of a sentence, using matrix-vector operations. This method works for sentence sentiment analysis because it depends on parsing. On the other hand, according to Le and Mikolov (2014) [111], this approach of simply averaging the vectors from word2vec performs poorly for sentiment analysis tasks, because it "loses the word order in the same way as the standard bag-of-words models do" and "fail[s] to recognize many sophisticated linguistic phenomena, for instance, sarcasm".

To get a sentence vector for some known data (see Le and Mikolov (2014) [111], AM Dai et al. (2015) [56]). Researchers are also looking for the output of certain layers in RNN or LSTM network, recent example is (Mueller and Thyagarajan (2016) [145]) where they present an adaptation of LSTM network for labeled data comprised of pairs of variable-length sequences. Their model is applied to assess the semantic similarity between sentences. For the gensim doc2vec, many researchers could not get good

results, to overcome this problem, Lau and Baldwin (2016) [109] analyzed several neural network designs and their variations. For more information, see Lan and Xu (2018) [108].

1.2 Deafness and Text Simplification

The subject of this thesis was first suggested by the startup Datalink to help deaf people to help them to read complex institutional websites. The idea was to propose a "simplification" algorithm which could be applied to text which would be diagnosed as complex. Unfortunately, no large written corpora are available for deaf people making the task difficult in a first approach. Nevertheless, we explain below the issues of such simplification tasks from a social point of view and why simplification is an important linguistic issue.

1.2.1 Deafness

The statistics of deafness in France on 08/02/2021. According to the OMS, nearly one million children are born each year with deafness. In France, 6% of the 15-24 years old are concerned with incapacitating auditory deficiency (hearing loss higher than 40 dB for an adult and 30 dB for a child), as well as 65% of the population aged 65 years and more. The global aging of the French population has shown the shortcomings of the management of hearing impairment. Unlike visual impairment, it is poorly diagnosed, poorly corrected, and, above all, poorly effective according to the opinion of those treated. However, hearing loss has important repercussions on daily life.

There are several reasons why we would want to simplify texts and make them easier to comprehend, as there are potentially many people who could benefit from automatic text simplification. Among potential beneficiaries, we can mention several groups of people who have difficulties in reading texts (Belder and Moens (2012) [10], Aluísio and Gasperin (2010) [4]) such as children or young readers, non-native speakers of the language (including students and second language learners) (Belder and Moens (2012) [10]), readers with low literacy (Peterson (2007) [156]), such as people with dyslexia or aphasia (Devlin (1998) [60], Shewan and Canter (1971) [178]), deaf and hard of hearing persons, adults who have suffered a brain injury. There is evidence with considerable proof that manual text simplification is an effective support for many readers (Shewan (1971) [178], Devlin (1998) [60], Aluísio (2010) [4], Belder (2012) [10], Peterson (2007) [156], Max (2005) [133]). Automatic text simplification has only become a recognized research topic in the last few years. In the same spirit, Shewan & Canter (1971) [178] studied sentence comprehension in three groups of aphasic people (Broca's, Wernicke's, and amnesic), and a group of "normal people" as a baseline. They were able to establish that there is significant variability in comprehension ability among people with aphasia, with the Wernicke's performing the worst. They noted that syntactic complexity was the most difficult parameter for all subjects in their study. According to them, the more difficult the sentence is, the weaker its comprehension becomes. They also discussed the clinical and medical applications of their test.

Furthermore, concerning deafness, a study by Swanwick and Watson (2005) [185] on the early literacy of deaf children raises fundamental interrogations about their access to language, their early interaction experiences, and their literacy development. Still, we understand very little about how a young deaf child develops literacy skills, given their unique language situation. Additionally, it has been shown that deaf children encounter many reading difficulties resulting from the experiential and language deficit in early childhood (Swanwick and Watson (2005) [185], Kelly (1996) [100], Alegria (2004) [2]), as they typically learn to read with poor cognitive development as well as insufficient language skills (Quigley and Paul (1984) [162] [160], Marschark and Spencer (2010) [132]). Moreover, Kelly (1996) [100] and Alegria (2004) [2] have reported several studies that show that it can be complicated for deaf and hard-of-hearing readers to take full advantage of their vocabulary knowledge until they have reached a correct level of syntactic competence. Jesus Alegria argues that the reading and spelling abilities of deaf people are low and that reading achievement is generally low in this population. She reports that the theory on reading acquisition and reading disorders in the case of hearing people includes two levels, a first

linguistic level (morphological, lexical, syntactic, and semantic) and a second level of general knowledge about the world. Indeed, it has been proven that deaf and hearing-impaired persons, in general, have reading and comprehension problems.

During the last 30 years, many researchers have tackled the subject, such as the work of Quigley and his co-workers at the University of Illinois which has gained increasing recognition and influence in the UK since the mid-1970s. The Syntactic Ability Test, which was developed from detailed analyses of deaf children's knowledge of the syntactic structure of written English, has been the primary vehicle for bringing their research to the UK. The issues and objectives, design, development, and use of this test are highlighted in the following (Quigley and King (1985) [159]). Here is the history of related works and studies that have been conducted from the 1970s until now, in chronological order. First Hammermeister (1971) [85] analyzed the reading abilities of a group of deaf adults up to thirteen years beyond school-leaving age and realized that reading comprehension did not improve, despite vocabulary improvement and increase. Studies conducted in the 1980s and 1990s confirm these observations. Then, Quigley et al. (1977) [161] reported that 10-year-old children have difficulties with all the complex structures such as passive voice and relative clauses, coordination, subordination, pronominalization and that once they are 18 years old, they become more able to understand these constructs and concepts. Despite this, they still have significant difficulties with subordinates and relative clauses. During the same year, Trybus and Krashmer (1977) analyzed a sample of 1,000 deaf or hearing-impaired persons aged 20 years and older and obtained results that are consistent with previous research and quite similar to those obtained afterward by Conrad (1979) [50], who conducted a rigorous and meticulous study with almost all pupils leaving a special school for hearing-impaired children in England and Wales from 1974 to 1976. He investigated the reading level achieved by this population through the use of the Wide-span Reading Test (For a series of sentence pairs, this test consists of filling in the missing word in a sentence from the second sentence of the same pair, see Brimer (1972) [29]). The median reading age he got for the entire population was 9 years. For persons who present hearing losses greater than 86dB, about 50% of the students were totally illiterate (reading age equal to 7 years which is the zero level of the test). Finally, even if we assume that a functional level of literacy is not achieved until a reading age of 11±12 years, less than 15% percent of this population reached this level. In addition, a reading level limit is reached around the third grade (see also Paul & Jackson (1994) [154], Alegria (2004) [2] and Quigley et al. (1977) [161]). Robbins & Hatcher (1981) report that constructions that affected the most comprehension are passive voice, relative clauses, conjunctions, and pronouns. As a result, Allen et al. (1986) [3] found that the difference between deaf and hearing children tends to increase as a function of time (see also Harris (1994) [87], and Mahapatra and Sabat (2016) [128]).

Therefore, the problem has been massively studied, and they have all demonstrated, in different ways, that deafness clearly impacts the reading and comprehension of texts. Regarding readers with low literacy Max (2005) [133], several studies have shown that the difficulties encountered by deaf people are mainly syntactic rather than lexical, especially for reading (Shewan and Canter (1971) [178], Max (2006) [134]). In the beginning, in order to answer the problem, the focus was more on how to write and produce texts, by proposing writing guides for texts intended for deaf and hearing-impaired persons, these guides were destined for readers (children or students) as well as for authors (teachers, public organism, etc.). Meanwhile, several projects have been carried out regarding digital accessibility, such as subtitling films and series, for example, "Subtitling for Deaf Children on British Television" (see Zárate (2008) [214]) who carried out a study on how to cut out the lines as well as the speed of display and the words to be used and omitted using a corpus based on two episodes taken from Nine cartoons from five different British channels.

From this, we see that at the beginning the works were more focused on reading acquisition in the framework of language learning, such is the case with Peterson, who argues that simplifying can be helpful for teachers as well as for students. In their paper Petersen and M. Ostendorf. (2007) [156], they build two tools:
1) for a given topic and specific reading level, the first tool allows one to find the appropriate and adapted texts on the web. Thus, it matches the specific reading level with the appropriate documents in a given topic. In other words, it allows an automatic selection of a corpus of texts adapted to the

reading level of the concerned public (topic and type of texts). Since finding material resources adapted to students who are (*Limited English Proficient*) always remains a difficult task.
2) By analyzing articles that have been manually simplified (each article is in a pair, an original version and a simplified version), they extracted what people do most frequently when adapting a text, and then built the second tool. Given an intermediate-level text, the second tool simplifies it in terms of grammatical construction and sentence length. Other works were done to answer the problem in the context of education, such as J. Burstein, et al who built an automated text adaptation tool in several versions (v1 & v2, see Burstein (2007) [32]). This tool aims to perform one of the processes done by a teacher while practicing text adaptation to help with reading comprehension and English language skills development for English language learners. This processing includes text summaries, and vocabulary support (e.g., providing definitions, translations, synonyms, etc.). Even if these practices are time-consuming, since they involve modification of texts to make them more understandable given a student's reading level, they remain crucial since reading-level appropriate texts are hard to find. The feedback they got about the development of the tool from an educational perspective came mainly from teachers. Then, Max (2005) [133] addressed in his article the problem of the comprehensibility of texts and in particular the need to simplify the syntactic complexity of sentences for language-impaired readers (readers with comprehension disorders). He presents an approach based on manually developed simplification rules and their integration into a text processor. This allowed them to interactively validate the simplified sentences produced by the system, and thus to integrate this text simplification task into the authoring (the authors' text production) task.

Finally, around the year 2010 the simplification of texts for this specific population aimed to facilitate digital accessibility for audiovisual (TV, cinema, etc.) as well as for the web (administrative sites, etc.). Among the numerous works that have aimed to include deaf and hearing-impaired persons in this sense, we can mention Soledad Zárate (2008), who has worked on accessibility in audiovisual, especially the task of subtitling. In a first work, Zárate (2008) [214] considers the extensive and exhaustive research that has been conducted on the reading characteristics of deaf children in the field of deaf studies, and the limited research that has been conducted in the field of audiovisual translation by focusing on subtitling for deaf persons. In their paper, they discuss the practice of subtitling children's programs on British television. Nine cartoons from five different British channels have been considered for the study. Two subtitled episodes of each have been recorded and the following factors were carefully examined: segmentation, degree of editing, reading speed, typographical cues, and use of non-standard language. Later, Zárate (2010) [215] studied deaf children's reading characteristics and abilities in order to reduce the gap between audiovisual translation and Deaf studies in the context of the production of appropriate subtitles for the deaf and the hard-of-hearing persons, which requires a clear understanding of the intended target audience. Concerning digital accessibility to websites and the Internet, Aluisio and Gasperin (2010) [4] developed text adaptation tools for Brazilian Portuguese to enable and promote digital inclusion and accessibility (see also Candido et al. (2009) [33]). These tools are intended for both people with low literacy skills and authors that want to produce texts for this specific audience.

To summarize, there is evidence from studies using manually simplified texts that reading comprehension can be improved for readers with poor literacy by substituting difficult words, splitting long sentences, making discourse relations explicit, avoiding pre-posed adverbial clauses, and presenting information in cause-effect order. Such studies provided the early motivation for text simplification as a comprehension aid. However, instead of focusing specifically on deafness, we will instead stay within a general framework that includes all of the cases mentioned above of language-impaired readers, involving people with comprehension disorders or those with reading problems. The only difference here is that the focus is more on digital accessibility and not on language learning.

After presenting the reading and comprehension problems related to deafness and the advantages of text simplification to face these problems, I will first talk about automatic text simplification in a general framework, give some history, and heuristics, and present the different ways in which the problem has been approached, both in theory and in practice. Then I will talk about different simplification techniques such as improving readability and/or understandability, and performing lexical and/or syntactic simplification. Later, I will discuss a bit about semantic simplification.

1.2.2 Text Simplification techniques

History and general definition of text simplification

The way of defining simplification and the way of constructing a corpus have both evolved during the last few years. In short, Text Simplification (TS) is the process of editing natural language in order to reduce its complexity and improve its readability and understandability. In other words, TS consists of editing an input text into a version that is less complex linguistically which may require syntax modifications or lexicon substitutions, or both, in order to improve the comprehensibility of the language for an end user. The choice of style (formal, informal, popular, etc.) differs from one context to another. This is a very important aspect of the presentation and communication of the information. Presenting it simply (as opposed to in a complex way) is always important and beneficial. As stated by Leon Tolstoy, "There is no greatness where there is not simplicity, goodness and truth" (see Yatskar et al. (2010) [210]). Systems that can transcribe the text into simple versions offer the potential to make information available to a wider audience as discussed above, including non-native speakers, children, laypeople, and so on (see Yatskar et al. (2010) [210]). However "The art of simplicity is a puzzle of complexity." as Douglas Horton affirmed, and the evaluation of TS results still remains problematic. In this second section, we will start by giving some general definitions to briefly cover the vocabulary used on TS in order to become familiar with these terms and notions which are specific to linguistics. Afterward, we will clarify more technically what TS means further in the section, especially in terms of mathematical and statistical modeling. All text simplification rules defined below could be handly crafted as was done previously, however, the aim of this thesis is to automatically process all these tasks (Belder and Moens(2012) [10], Yatskar et al. (2010) [210], Ligozat et al. (2013) [124], Biran et al. (2011) [19]).

Simplification can be used for many applications, including second language learners such as mentioned above, by helping end users access relevant information, which would be too complex to understand if left unedited (Devlin (1998) [60], Caroll et al. (1998) [36], Alegria (2004) [2], Max (2005) [133], Max (2006) [134], Aluísio and Gasperin (2010) [4], Mahapatra and Sabat (2016) [128]). Thus, simplification is an important task (Max (2005) [133]), and although these different groups of people find texts difficult for many reasons, the causes of the difficulty in understanding these texts cover a broad horizon. What makes a sentence difficult is generally attributed to one or both of the following reasons: lexical difficulty (i.e. difficult words and sequences), syntactic difficulty (i.e. complex grammatical constructs). Automated simplification can also be considered a preprocessing tool. First, researchers in information theory, mainly linguistics, statistics, and more recently data science, used this TS more as a pre-processor for other tasks such as in pipelines and assistive technologies. Thus TS has been investigated mostly as a preprocessing step with the goal of improving NLP tasks, such as parsing (see Chandrasekar and Srinivas (1997) [38], Siddharthan (2006) [180], Jonnalagadda and Gonzalez (2009) [99]), semantic role labeling (see Vickrey and Koller (2008) [201]) and summarization (see Blake et al. (2007) [20]).

The number of publications in the field of automatic text simplification (ATS) has been increasing continuously during the last few years. The graph on the figure (1.7) represents the number of all results obtained on ATS in *Google scholar*[34] research for each year. Sometimes there are duplicates as for articles published in a journal but also in a conference or a book. Still, this allows us to have an idea of the evolution of the field.

[34] https://scholar.google.com/scholar?q=automatic+text+simplification&hl=fr&as_sdt=0%2C5&as_ylo=2021&as_yhi=2022

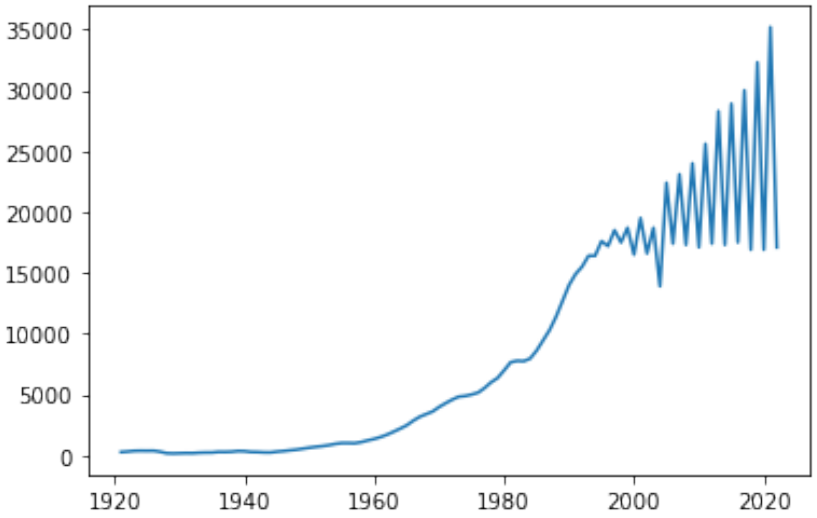


Figure 1.7: Number of *Google scholar* results in Automatic Text Simplification between 1920 and 2022.

Text simplification is closely related to machine translation (MT) even though these two tasks are distinct. Although TS could be described as a special case of MT, it differs in spirit since simplification must find the trade-off between preserving meaning (central to paraphrasing) and reducing complexity (no consideration for paraphrasing). Simplification can also be considered as a particular form of MT in which the two languages in question are strongly related, or are the same with additional constraints on the target language (the language into which one translates, as opposed to the source language from which one translates) see Yatskar et al. (2010) [210].

In the first part, we will focus more on the linguistic aspects behind simplification. Two subtasks are generally distinguished in automatic textual simplification, although they are not totally disconnected: syntactic simplification (SS) and lexical simplification (LS). We will first talk about the division of the problem into these two main categories, LS and SS. There are many other approaches to the simplification task, including statistical machine translation and hybrid techniques. Since text simplification is a non-trivial task that is rapidly growing into its own field (see Belder and Moens (2012) [10]). We will also present briefly another well-known way to cut the problem, the one that divides simplification into two parts, improving *readability* and improving *comprehensibility*. A sentence with good readability, is such that at each position of the sentence, the initial word and everything that follows is still in the reader's memory. So readability is often considered as the ease of reading a sentence and the speed with which the previously read information is retained. A sentence with high comprehensibility (score, rank, etc.) is such that each sequence it contains is immediately and quickly understood after its reading. In other words, comprehensibility is explained by the ease with which one understands the meaning of a sentence and the simplicity of capturing the semantic information it contains. However, we can see that these last two definitions are overlapping, in the sense that it does not help to do one without the other, whereas splitting the simplification task into lexical and syntactic would allow to put the theories into practice while offering an improvement of readability (by simplifying syntax and lexicon) as well as comprehensibility (mainly by using syntactic simplification). In the second section, we will focus more on the statistical aspects behind simplification. In the next paragraph, we'll not distinguish between readability and understandability, but we'll instead treat only text comprehension that overlaps both readability and understandability and then consider text simplification in a lexical or syntactic level aiming to improve text comprehension in a large sense, including all of readability and understandability of any type of end users (teacher, student, internet user lambda, etc.).

History of Lexical simplification

Lexical simplification (LS) is a particular case of text simplification. In brief, LS is the task of identifying and replacing complex words with simpler substitutes. The point is not to simplify the grammar of a text, but rather to focus on simplifying complex aspects of vocabulary (Belder and Moens (2012) [10]). Early work on lexical simplification involved replacing words with more common synonyms from WordNet or other dictionaries (Devlin (1998) [60], Devlin (1999) [61], Caroll et al. (1998) [36], Caroll et al. (1999) [37]). Lexical complexity is usually estimated in terms of word length (number of characters) or number of syllables on the one hand, or word frequency on the other hand, based on corpus analysis or a database. Drndarević and Saggion (2012) [65] showed that word frequency and word length in terms of the number of characters or syllables were useful indicators of lexical complexity from a Spanish parallel corpus (Ligozat et al. (2013) [124], Bott et al. (2012) [24]). After more than a decade since the first appearance of publications on its practical implementation in the literature (Devlin (1998) [60], Caroll et al. (1998) [36]), Lexical Simplification is receiving a renewed interest (McCarthy and Navigli (2007) [136], Zhao et al. (2007) [217], McCarthy and Navigli (2009) [137], Yatskar et al. (2010) [210], Aluísio and Gasperin (2010) [4], Biran et al. (2011) [19], Belder and Moens (2012) [10], Saggion et al. (2015) [171]).

It is important that the original meaning of the input text is not altered, and that it remains fluid (Belder and Moens (2012) [10]). In order to make such substitutions, it is important to first identify equivalent words that match the context, and then choose the simplest word. Despite the fact that this task seems straightforward and simple, the evaluation algorithms that allow it are not since it is a hard problem to evaluate. Indeed, simplifications are context-dependent, and therefore generating a complete list of simplifications is very difficult. The reason is that on the one hand, it is difficult to build a database that contains an exhaustive list of words that are easier to understand in different contexts, and on the other hand having an absolute order for this list of synonymous expressions is very complicated (Belder and Moens (2012) [10]). Simplification can also be seen as a translation task between a standard language and a simplified version of that language; however, it is important to note that in classical translations it is difficult to produce fully parallel corpora (Ligozat et al. (2013) [124]).

Here are some examples of sentences with alternative words. (results obtained by Belder and Moens (2012) [10]):

- Rabbits often feed on young, *tender* perennial growth as it emerges in spring, or on young transplants. [[soft], [tender, delicate]]
- Performance test for a system coupled with a locally manufactured station engine model MWM will start *shortly*. [[shortly, soon], [before long], [presently]]
- Perhaps the effect of West Nile Virus is sufficient to extinguish endemic birds already *severely* stressed by habitat losses. [[highly], [seriously, severely, extremely], [gravely], [critically]]
- Mutual Funds are so *severely* conflicted that they will not avail themselves of the alleged benefits of the proposed rule. [[badly], [seriously, severely, heavily], [extremely, gravely]]

Finally, some examples of simplifications found by the methods used by Yatskar et al. (2010) [210]:

"stands for" → "is the same as", "indigenous" → "native", "permitted" → "allowed",
"concealed" → "hidden", "collapsed" → "fell down", "annually" → "every year".

In studying the SemEval 2012 corpus, Ligozat et al. (2013) [124] considered simplification in terms of characterizing the simplicity of lexical items in context. This characterization includes criteria concerning the item itself, the local context of the item, and the more general context of the item (see also François and Fairon (2012) [72], Jauhar and Specia (2012) [94], Specia et al. (2012) [183]). The criterion concerning the element itself is mainly derived from measures of text readability such as the size of the element in terms of the number of characters or syllables, the frequency of the element in the corpus, the presence of this element in lists of simple words or others coming from psycholinguistic

characteristics (such as concreteness, age of acquisition, etc.) The local context of the element, especially in the case of membership in a collocation, gives information on the adequate substitute. As for instance the sequence "pay attention" where "pay" means "give" and not "to make a payment" by "giving the money", or "fast food", where it's not the food that's fast, but rather its preparation. The more general context of the element, such as its thematic context. Ligozat et al. (2013) [124] assume that the use of a larger context better captures the semantic specificities of the substitutes and the linguistic environment in which these substitutes evolve.

History of Syntactic Simplification

Syntactic simplification is typically done in three phases as shown in the Figure (1.8). First, the text is analyzed to identify its structure and parse tree. This may be done at varying granularity but has been shown to work at a rather coarse level. At this level, words and phrases are grouped together into 'super-tags' which represent a chunk of the underlying sentence. These super-tags can be joined together with conventional grammar rules to provide a structured version of the text. During the analysis phase, the complexity of a sentence is determined to decide whether it will require simplification. This may be done by automatically matching rules but has also been done using a support vector machine binary classifier (Gasperin et al. (2009) [76]). While POS taggers only indicate the grammatical role of a particular word, a parse tree represents the syntactic structure of the whole sentence, giving complete details on how the words in it are related to each other. Sentence simplification systems6,10,11 usually have parsers as an integral part of their algorithm, while there are few systems that use only POS. The latter aims for fast simplification at the point of application, while the former gives higher importance to the accuracy of the output (Jonnalagadda and Gonzalez (2010) [99]). Chandrasekar and Srinivas (1997) [38], for example, use an architecture with two stages – analysis and transformation. There are various discourse-level issues that arise when carrying out sentence-level syntactic restructuring (Siddharthan (2006) Siddharthan (2006) [180]).

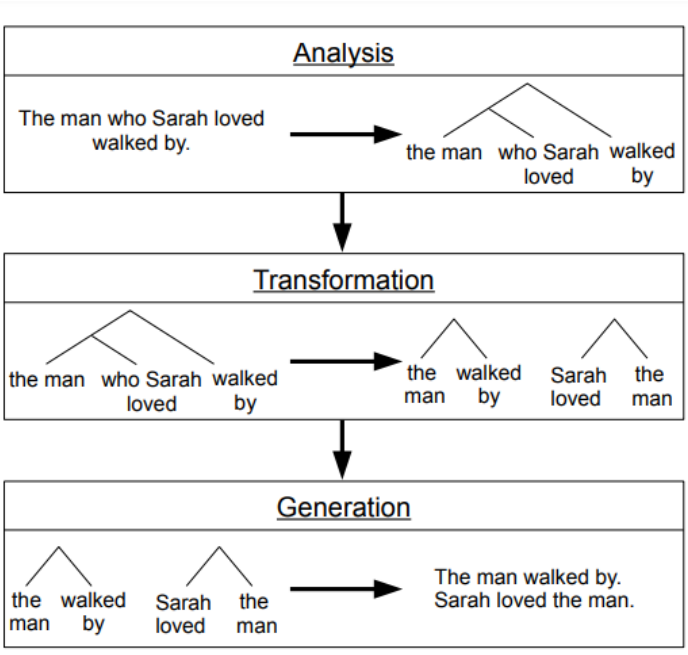


Figure 1.8: The syntactic simplification (Shardlow (2014) [177])

The best-performing tool existing nowadays is the following called Grammarly[35] that was launched in 2009 by Alex Shevchenko, Max Lytvyn, and Dmytro Lider.

[35] https://www.grammarly.com/

Conclusion

Initially, we projected to use almost all existing NLP techniques as pre-processing, such as chunking and parsing, before starting the modeling or implementation of a text classification or automatic translation model from complex to simple text. However, we ended up using only tokenization, stemming, and POS tagging mainly for two reasons: 1) The need to manually input the grammar rules for chunking or parsing made the task difficult to automate. 2) The high performance of new neural network models, which do not require pre-processing (only tokenization and stemming).

Therefore, the only tools that we are going to use are POS tagging and stemming. Regarding simplification, we did not choose a specific method among lexical simplification, syntactic simplification, and hybrid simplification. Instead, we opted for an SMT (Statistical Machine Translation) approach, which includes all the above approaches, depending on the context. This can be explained by the fact that we will use the Wikipedia encyclopedia as a training dataset, and a simplified version (simple English Wikipedia) which already contains both lexical and/or syntactical simplified articles. Selecting just one method separately from the others could be a mistake, especially if we aim at producing or generating texts. The POS tagging was also reprogrammed to test the mathematical models developed in the next chapters.

Chapter 2

GEL and Complexity Measure

In this chapter, we combine methods used in penalized generalized empirical likelihood frameworks (Owen (1988) [151], Owen et al. (1990) [150]) with feature extraction techniques that enable projecting textual data into numerical spaces. We first introduce methods that can be used to generate classifiers for POS tagging problems utilizing feature-based models. The key to the proposed technique is the combination of a feature extraction stage composing the data in an "elementary space" to produce features, and a classification stage utilizing a Generalized Empirical Likelihood framework (GEL) to estimate the model (i.e. to learn the weight of each feature extracted). We relate this approach to the Maximum Entropy principle (Ratnaparkhi et al. (1996) [166]) used in Natural Language Processing (NLP). Since the features belong to a large dimensional space we propose a penalization method based on the dual representation of the original problem. Finally, the same approach will be used to construct a classifier, which gets a sentence and returns a binary output that indicates whether the input sentence is complex "0" or simple "1". We will also discuss more recent techniques used in GEL (Bertail et al. (2006, 2007, 2015) [13, 17, 16]) when the number of features gets very large in comparison to the sample size.

Introduction

Standard learning systems (such as neural networks or decision trees) work on input data after they have been transformed into "feature" vectors (descriptors, characteristics or attributes) $X_1, \ldots, X_p \in \mathcal{X}$ from an n-dimensional space. However, there are cases where input data cannot be easily described by explicit feature vectors: for example, images, graphics, text documents, bio-sequences and algorithms in source code form. For such data sets, constructing a feature extraction module can be as complex and costly as solving the problem as a whole. One of the effective alternatives to extracting these explicit features is provided in this chapter. We refer to Collins (2002) [47], Guyon et al. (2008) [84], Ratnaparkhi et al. (1996) [166], Yogatama (2015) [211].

We adopt a generalized maximum entropy approach since it allows us to include diverse and varied sources of information, without causing fragmentation, while allowing us to avoid assumptions about predictors (or explanatory variables), particularly independent ones. In addition, it allows the incorporation of other techniques and hypotheses like those of hidden Markov models. The maximum entropy method was applied in POS tagging previously by Ratnaparkhi et al. (1996) [166]. Then, several other authors studied this method in the following years to explore its effectiveness on other tasks such as named entity recognition speech tagging. See the paper of Borthwick et al. (1998) [21], which describes the statistical system known as named-entity recognition where they used a maximum entropy model. They qualify it as a flexible tool that is able to make use of a diverse range of knowledge sources in making decisions.

The first section presents the studied method (MaxEnt) that will represent a reference for the

rest of the study. For the remainder of the chapter, "MaxEnt" will refer to Maximum Entropy approach combined with Feature-based Models. We will focus on POS tagging and review some of its standard facts and indicate how these techniques may be used in natural language processing in order to reduce text complexities, improve their readability and comprehension and then (perhaps) detect some regularities in the syntactic structure between simple and complex texts. In section 2, we summarize without proof the relevant material on mathematical methods, by first setting up notations and terminology and then introducing the notion of Feature-based models.

Section 3 is intended to motivate our investigation of log-linear models. This section is devoted to the study of Feature-based models. It provides a detailed exposition of Generalized Empirical Likelihood (Owen (1988) [151], Owen et al. (1990) [150]) and establishes the relation between generalized empirical likelihood and parametric likelihood settings (Bertail (2006) [13], Bertail et al. (2015) [16]). In the fourth section, we indicate how these techniques may be used to classify texts into two classes, simple and complex. This last section also presents some preliminaries to POS tagging in order to construct a computer tool that automatically processes this task.

2.1 Part Of Speech Tagging

POS tagging is the linguistic process of assigning each word in a sentence the part of speech that it assumes in that sentence. By part of speech we mean the corresponding grammatical information such as the part of speech, gender, number, etc. So, for a given *input*, such as a sequence of words plus a tagset, we seek to predict the correct output (a single best tag for each word), automatically by using a computer tool. The main problem of POS tagging is to resolve ambiguities, by adequately choosing the proper tag for the context. Thus, the choice of the POS tagging task is justified by the ease of checking how many tags have been correctly predicted. So we may immediately notice that this automatism can be done in a more efficient way with methods that look at the local context (see the subsection 2.1.5).

2.1.1 Advantages of POS tagging

It is a simple task (usually linear processing time), which can be used in many other applications. As Anna Feldman (2010) [68] said, Part-of-speech tagging is important for a variety of reasons. For example, corpora that have been POS tagged can be very useful in linguistic research to detect examples or frequencies of particular constructions in large corpora (e. g. Meurers (2005) [141]). POS information can also serve as a basis for syntactic parsing. Knowing the part of speech information for each word in an input sentence helps to determine the correct syntax for a given formalism. Thus, it can be used as a preprocessor for a parser (which speeds up the parsing). Additionally, automatic POS taggers can help in building automatic word-sense disambiguation algorithms. Furthermore, knowing which POS occurs next to which can be useful in a language model for text production or morphological generation (i.e. generating words and phrases, mapping a linguistic stem to all matching words.. etc.). This knowledge is crucial for extracting verbs or other important words from documents, which later can be used for text summarization or simplification.

2.1.2 Tagsets and Examples

There are several tagsets, such as the ones mentioned in Chapter 1: Penn Treebank[1], Brown[2], and British national corpus[3]. The POS tags used below are taken from the tagset Penn Treebank Corpus, proposed at the University of Pennsylvania that includes 36 tags, see Figure 2.1. Recall that there is also a collection called *universal tagset*[4], which just says if the word is a **pron, noun, verb, det, adj, adv, or punct**. So it contains 7 tags in total.

[1] https://web.archive.org/web/20131109202842/http://www.cis.upenn.edu/~treebank/
[2] https://web.archive.org/web/20080706074336/http://www.scs.leeds.ac.uk/ccalas/tagsets/brown.html
[3] https://ucrel.lancs.ac.uk/claws7tags.html
[4] https://universaldependencies.org/u/pos/

CC	Coordinating conj.	TO	infinitival *to*
CD	Cardinal number	UH	Interjection
DT	Determiner	VB	Verb, base form
EX	Existential there	VBD	Verb, past tense
FW	Foreign word	VBG	Verb, gerund/present pple
IN	Preposition	VBN	Verb, past participle
JJ	Adjective	VBP	Verb, non-3rd ps. sg. present
JJR	Adjective, comparative	VBZ	Verb, 3rd ps. sg. present
JJS	Adjective, superlative	WDT	Wh-determiner
LS	List item marker	WP	Wh-pronoun
MD	Modal	WP$	Possessive *wh*-pronoun
NN	Noun, singular or mass	WRB	Wh-adverb
NNS	Noun, plural	#	Pound sign
NNP	Proper noun, singular	$	Dollar sign
NNPS	Proper noun, plural	.	Sentence-final punctuation
PDT	Predeterminer	,	Comma
POS	Possessive ending	:	Colon, semi-colon
PRP	Personal pronoun	(	Left bracket character
PP$	Possessive pronoun	)	Right bracket character
RB	Adverb	"	Straight double quote
RBR	Adverb, comparative	‘	Left open single quote
RBS	Adverb, superlative	“	Left open double quote
RP	Particle	’	Right close single quote
SYM	Symbol	”	Right close double quote

Figure 2.1: PennTreebank tagset. See A.Taylor & al. (2003) *[188]*

POS tagging examples Here are two examples of POS-tagged sentences:

"I saw a girl with a telescope." ; "The grand jury commented on a number of other topics."

I	saw	a	girl	with	a	telescope	.
↓	↓	↓	↓	↓	↓	↓	↓
PRP	VBD	DT	NN	IN	DT	NN	.

The	grand	jury	commented	on	a	number	of	other	topics	.
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
DT	JJ	NN	VBD	IN	DT	NN	IN	JJ	NNS	.

2.1.3 Mathematical model: supervised learning

Approaches There are now numerous systems for the automatic assignment of parts of speech tagging, employing many different machine learning methods. Among recent top-performing methods are Hidden Markov Models (Brants (2000) [26]), maximum entropy approaches (Ratnaparkhi et al. (1996) [166]), and transformation-based learning (Brill (1994) [28]). An overview of these and other approaches can be found in Manning and Schoetze (1999) [130] (Chapter 10). Notice that Ratnaparkhi's thesis (1998) [165] contains different NLP tasks like sentence boundary detection (sentence tokenization), POS-tagging, and parsing where he used ideas similar to the maximum entropy method.

Mathematical modeling POS tagging task can be considered as a ***classification*** problem, where the goal is to estimate a function $g : \boldsymbol{\mathcal{X}} \to \boldsymbol{\mathcal{Y}}$ which maps an object $x \in \boldsymbol{\mathcal{X}}$, where $\mathcal{X}$ is some abstract measurable space (texts, sentence, sequence of words) equipped with a σ-algebra, to its correct class $y \in \boldsymbol{\mathcal{Y}}$. That is to say we consider a classifier of the form

$$\begin{cases} g : & \boldsymbol{\mathcal{X}} \longrightarrow \boldsymbol{\mathcal{Y}} \\ & x \longmapsto y. \end{cases}$$

In pattern classification problems the goal is to guess or predict the unknown class of a given observation, based on a sample of n independent observations $z_1 = (x_1, y_1), ..., z_n = (x_n, y_n)$. An observation is often a collection of numerical or categorical measurements represented by a q-dimensional vector x, in some cases, it could be a curve or an image, or a text (or even a video, especially with

neural network classification models). The unknown nature of the observation is called a class and it is denoted by y. The classifier makes an error on x if $g(x) \neq y$.

For the formalization of the learning problem, we consider a probabilistic setting and a pair of random variables $Z = (X, Y) \in \mathcal{X} \times \mathcal{Y}$ representing the observation and its corresponding class. In this configuration, the distribution of random pair $Z = (X, Y)$ may be described by the distribution of $\mathcal{X}$ (given by the probabilities $\mathbb{P}[X \in A]$ for all subsets A of $\mathcal{X}$) and by $\eta_y(x) = \mathbb{P}[Y = y | X = x]$ for all $y \in \mathcal{Y}$. The functions η_y are called the a posteriori probabilities. The performance of the classifier g is then measured by its probability error $\text{Loss}(g) = \mathbb{P}[g(X) \neq Y]$ (see Boucheron et al. (2005) [25]).

Given η_y, one may easily construct a classifier with minimal probability of error

$$g^*(x) = \arg\max_{y \in \mathcal{Y}} \{\eta_y(x)\}.$$

In particular, it is easy to see that if we consider the case of $y \in \{0, 1\}$ and

$$g^*(x) = \begin{cases} 1 & \text{if } \eta_1(x) > 1/2, \\ 0 & \text{otherwise.} \end{cases}$$

then $\text{Loss}(g^*) \leq \text{Loss}(g)$ for any classifier g since

$$\text{Loss}(g) - \text{Loss}^* = \mathbb{E}\left[\mathbf{1}_{\{g(X) \neq g^*(X)\}} |2\eta_1(X) - 1|\right] \geq 0,$$

where Loss^* represents the minimal risk defined by $\text{Loss}^* = \text{Loss}(g^*)$. Loss^* is called the *Bayes risk* or *Bayes error* (see Devroye et al. (2013) [62] pg 20-21).

In the same spirit, we can model the POS tagging task using a similar setting. Recall that the goal of POS tagging is to find the best tag sequence $\hat{y} = t_1^*, ..., t_N^*$ for a given sentence $x = w_1, ..., w_N$ (containing N words). That is to say for a given sentence or word sequence, pick the best tag for each word (the best here corresponds to the most correct with a meaning that is consistent with the grammatical structure of the whole sentence). The most common approach is to build a model p that can be considered as a probability distribution where either the conditional probability is maximized

$$t_1^*, .., t_N^* = \arg\max_{t_1, ..., t_N \in \mathcal{T}} \left[p\left(t_1, .., t_N \mid w_1, .., w_N\right)\right], \tag{2.1}$$

or alternatively, the joint probability is maximized

$$t_1^*, .., t_N^* = \arg\max_{t_1, ..., t_N \in \mathcal{T}} \left[p\left(t_1, .., t_N, w_1, .., w_N\right)\right]. \tag{2.2}$$

Notice that the model $p(t_1, .., t_N, w_1, .., w_N)$ can be seen as a partial joint likelihood since it is a function that measures the goodness of fit of a statistical model to a piece from a sample of data for given values of the unknown parameters.

In practice, we have $\tilde{n}$ independent random variables in the form of pairs (containing the sentences and their corresponding tag sequence) having a distribution P in the torus $\bigcup_{k=1}^{\infty} \mathcal{T}^k \times \bigcup_{k=1}^{\infty} \mathcal{D}^k$ where $\mathcal{T}$ and $\mathcal{D}$ are respectively the tagset and the dictionary of words. Each sentence x_i contains a random number of words N_i. Let us denote $n = \sum_{i=1}^{\tilde{n}} N_i$ the total number of words.

Generally, N is a random number, because sentence sizes in terms of the number of words vary according to the situation. However, in this particular case of POS tagging, since the goal is to find one and only one tag for each word, i.e. the number of tags to guess is equal to the number of words, then the size of the sentence will not have any influence on the model. Thus, in all that follows, we will

assume that the sentences are independent of each other. As described in the paragraph (2.1.4), one of the most suitable ways to express such models is to write the likelihood of one sentence as

$$p\left(t_1, .., t_N \mid w_1, .., w_N\right) = \prod_{i=1}^{N} p\left(t_i \mid x_i\right),$$

where x_i is a function of the word w_i and the words surrounding w_i, and potentially some elements contained in w_i (such as affixes, etc.). Then, in order to predict the best tag sequence for a given new sentence, this formulation leads to the following final expression

$$t_1^*, .., t_N^* = \underset{t_1, ..., t_N \in \mathcal{T}}{\arg\max} \left[\prod_{i=1}^{N} p\left(t_i \mid x_i\right)\right].$$

In the next section, we will see how we can use Maximum entropy models to estimate these conditional probabilities, and for more convenience, we will simply first consider $p\left(y|x\right)$ where y could represent, in the example of POS tagging, a tag sequence and x the given sentence we aim to tag, or in the example of machine translation, y could represent the potential translations and x the word to translate. Notice that N does not represent the sample size but rather the size of a given sentence.

2.1.4 Basic models for POS-tagging and extensions

Now, instead of building a classifier g such that $g(x) = y$ which directly classifies sentences, we instead build a model such that for a given sentence s we have $\forall w_i \in s,\ g(w_i) = t_i$ so that concatenating the predicted tags for each word gives the sequence of tags for the entire sentence. As the model may not always succeed in deciding between two potential tags for the same word w_i, this can lead the classifier to always choose the same tag that co-occurs with this word w_i the most. In order to avoid this, the idea would be to consider that the classifier g looks not only at the word w_i itself but also at the sentence s that contains it, i.e. the words $w_{i-k}, ..., w_{i-1}$ and $w_{i+1}, ..., w_{i+k}$ that surround the word w_i to be tagged (in this example we considered a window of size $2k$ with k words at the left and k words at the right).

In the next sections, we will briefly cover some known statistical models used for POS tagging, from simple and naive models to more sophisticated/elaborated ones.

Notations The following notations are considered for the rest of this section :
$\mathcal{T}$: the tagset of size $|\mathcal{T}|$
D : initial database with $|D| = n$ where n is very large
D^M : the training (or learning) data for the M model (simply D or transformed D)
w_i : i^{th} word in a sentence (w_1, ..., w_N containing N words for example)
t_i : i^e tag in a sequence (t_1, ..., t_N containing N tags for example)
$\mathcal{D}$: the dictionary or the set of all unique words in D

Remark It is important to notice that $D \subsetneq \mathcal{D} \times \mathcal{T}$, because the corpora used in learning often do not contain all the possible combinations between $\mathcal{D}$'s elements (words) and $\mathcal{T}$'s elements (tags). Here we're only talking about combinations that are syntactically correct since we can find some $(w, t) \in \mathcal{D} \times \mathcal{T}$ which are not linguistically possible like ("*the*", VB), i.e. to assign the grammatical class "Verb" to the word "the", which is clearly invalid.

Bag-of-Words models (BoW)

We start with a Bag of Words (BoW) based tagger that uses the simplest (and naive) way. Initially, BoW is a technique used in document classification. Still, it can be applied too in some other problems,

like POS tagging. The tagger learns a conditional probability model from tagged text, using the independence assumption between words conditionally to their tags, ie. the events $(t_i \mid w_i)$ and $(t_{i+1} \mid w_{i+1})$ are independent for all $i \in \{1, \ldots, n\}$, which means that the probability of a word depends only on its tag. The database used for training the classifier in BoW models is noted D^{BoW} such as $D^{BoW} = \{(w_1, t_1), ..., (w_n, t_n)\}$. This is similar to the one-hot encoding mentioned in the first chapter (corpora subsection). So with these assumptions, we can write the objective function as

$$p\left(t_1^*, ..., t_N^* \mid w_1, ..., w_N\right) = \max_{t_1, ..., t_N \in \mathcal{T}} \left[p\left(t_1, ..., t_N \mid w_1, ..., w_N\right)\right] = \max_{t_1, ..., t_N \in \mathcal{T}} \left[\prod_{i=1}^{N} p\left(t_i \mid w_i\right)\right].$$

One can immediately remark that this model is not adapted to the problem since the words (with their tags) are closely related to the surrounding words. For instance, the word "*flies*" is a verb in the phrase "*she* ***flies***" but it is a noun in the phrase "*the* ***flies***" which perfectly demonstrates the dependency between the grammatical class of the word "*flies*" and the preceding word (or the context it is in).

Estimation:

The estimation is straightforward in this case, as it is sufficient to compute the empirical frequencies, i.e. for a given word w_j, the empirical probability of any tag t_i from the tagset conditionally to w_j is given by

$$\forall t_i \in \mathcal{T}, \quad \hat{p}\left(t_i \mid w_j\right) = \frac{freq\left(t_i, w_j\right)}{freq\left(t_i\right)},$$

$$freq(t_i, w_j) = \frac{1}{n} \sum_{k=1}^{n} \mathbf{1}_{\{t_k = t_i, w_k = w_j\}}(t_k, w_k) \; ; \; freq(t_i) = \frac{1}{n} \sum_{k=1}^{n} \mathbf{1}_{\{t_k = t_i\}}(t_k).$$

where $\mathbf{1}_{\{t_k = t_i\}}(t_k)$ is equal to 1 if $t_k = t_i$ and it is equal to 0 otherwise.

Notice that t_k is the k-th observation in the dataset, which means that there are many values of $k \in \{1 \ldots n\}$ for which t_k is equal to the same i-th tag t_i in the tagset $\mathcal{T}$. However for each two tags t_i and t_j from the tagset $\mathcal{T}$, if $i \neq j$ we have $t_i \neq t_j$. So, $freq(t_i, w_j)$ represents the number of times the word w_j was tagged by t_i, this number is scaled by the size of the database, and $freq(t_i)$ represents the number of times the tag t_i was observed in the database normalized by the database size to get the empirical frequency.

Thus for a new sentence $w_1, ..., w_N$, the predicted tag for each word is given as follows

$$\forall w_k \in [w_1, .., w_N], \quad \hat{t}_k = \arg\max_{t_i \in \mathcal{T}} \left\{\hat{p}\left(t_i \mid w_k\right)\right\}.$$

This model will only learn the most frequent pairs for each word and will always return the same tag as an output for that word, no matter the context where the word will appear. It is obvious that this model is not well adapted for texts where there are dependencies between consecutive words in the same sentence. Consider for example these two sentences : *Sentence*$_1$*: "I process something..."* and *Sentence*$_2$*: "Do the process for..."*. We can see that the word *process* is a verb in the first sentence while it is a noun in the second. If the word *process* has been observed as a noun more than as a verb, then this model will predict the tag "*noun*" for the word *process* in both sentences.

Markov Models (MM)

Markov models allow us to consider a little more dependency, i.e. the dependencies between every two tags (in 1^{st} order Markov model, or between every 3 tags for a 2^{nd} order MM, etc.).

$D^{Markov} = \{(w_1,\, t_1)\,,\, ...,\, (w_n,\, t_n)\} \cup \{(t_i, t_j)\,,\, \forall t_i, t_j \in \mathcal{T}\}$ where (t_i, t_j) is the event that t_i precedes t_j in the same sentence (or in the same tag sequence). Thus, in an MM, the aim is to maximize the joint probability (see Thorsten Brants (2000) [26]). We can write the joint probability as follows

$$p\,(t_1, ..., t_N, w, ..., w_N) = p\,(t_1, ..., t_N)\, p\,(w_1, ..., w_N \mid t_1, ..., t_N)\,,$$

and if we make a simplifying bigram assumption to approximate these two factors we get the following expressions

- Probability of a word depends only on its tag:

$$p\,(\mathbf{w_i} \mid w_1,\, t_1,\, ...\,,\, t_{i-1},\, t_i) = p\,(\mathbf{w_i} \mid t_i)$$

- Tag history approximated by two most recent tags (trigram: two most recent + current state)

$$p\,(\mathbf{t_i} \mid w_1,\, t_1,\, ...\,,\, t_{i-1}) = p\,(\mathbf{t_i} \mid t_{i-1})$$

First-order Markov Model Therefore, by using bigrams the probability of a tag sequence is calculated as follows

$$p\,(t_1, ..., t_N) = p\,(t_1)\, p\,(t_2 \mid t_1)\, ...p\,(t_N \mid t_{N-1})\,,$$

and the probability of word sequence (knowing a tag sequence) is given by the formula

$$p\,(w_1, ..., w_N \mid t_1, ..., t_N) = p\,(w_1 \mid t_1)\, p\,(w_2 \mid t_2)\, ...p\,(w_N \mid t_N)$$

Then, it follows

$$\begin{cases} p\,(t_1, ..., t_N, w, ..., w_N) & = p\,(t_1, ..., t_N)\, p\,(w_1, ..., w_N \mid t_1, ..., t_N) \\ & = p(t_1)\, \left[\prod_{i=2}^{N} p(t_i \mid t_{i-1})\right] \left[\prod_{i=1}^{N} p(w_i \mid t_i)\right]. \end{cases}$$

Estimation:
Denote

$$\left\{p_{i,j}^{\text{words}} = p\,(w_j \mid t_i)\right\}_{(i,j)\in[\![1,|\mathcal{T}|]\!]\times[\![1,|\mathcal{D}|]\!]}$$

and

$$\left\{p_{i,j}^{\text{tags}} = p\,(t_j \mid t_i)\right\}_{(i,j)\in[\![1,|\mathcal{T}|]\!]^2}$$

represent the transition matrix (or transition probabilities) with $\hat{p}^{\text{words}}$ and $\hat{p}^{\text{tags}}$ their respective estimates. Each component $\hat{p}_{i,j}$ represents an estimated probability of the event that the tag t_i precedes t_j where $nb(t_i, t_j)$ and $nb(t_i)$ are the number of occurrences of the tag pair (t_i, t_j) and the tag t_i respectively. Then we have for all $t_i, t_j \in \mathcal{T}$ the transition matrix as follows

$$\hat{p}^{\text{tags}} = \left\{\hat{p}_{i,j}^{\text{tags}}\right\}_{i,j\in[\![1,|\mathcal{T}|]\!]} \quad \text{where } \hat{p}_{i,j}^{\text{tags}} = \frac{nb\,(t_i,\, t_j)}{nb\,(t_i)}.$$

Denote the probability of the event that the word w is tagged by the tag t_i. We now can obtain the estimations of these probabilities in a similar way to the probability of tag transitions as above. We have for all $t_i, w_j \in \mathcal{T} \times \mathcal{D}$,

$$\hat{p}^{\text{words}} = \left\{\hat{p}_{i,j}^{\text{words}}\right\}_{(i,j)\in[\![1,|\mathcal{T}|]\!]\times[\![1,m]\!]} \quad \text{with } \hat{p}_{i,j}^{\text{words}} = \frac{nb\,(t_i,\, w_j)}{nb\,(t_i)},$$

where m represents the total number of the observed words w_j (with repetitions). Remark that each observation $\mathbf{obs}_k = (w_k, t_k)$ with $k = 1, \dots n$ can occur many times, while for each $(i, j) \neq (i', j')$ we have $w_j \neq w_{j'}$ and $t_i \neq t_{i'}$ so that the transition matrix makes sense by not having any repetitions. However, there might be a lot of pairs (w_j, t_i) that have never been observed in the dataset, they will certainly have a probability equal to zero, making the transition matrix sparse.

Problems in Markov Models Notice that the transition matrix $\{p(w_j \mid t_i)\}$ should be estimated for each couple $(t_i, w_j) \in \mathcal{T} \times \mathcal{D}$, even those that have not been observed. In general, one assumes independence between paires (w_i, t_i) and (w_{i+1}, t_{i+1}), $\forall i \in \{1, \ldots, n-1\}$, which is not necessarily the case in practice.

2.1.5 More advanced models

Feature based models

Let us remember that text corpora represent unstructured databases. This is why we use functions called features, which make it possible to organize corpora and extract (contextual) information useful for the processing considered (POS tagging eventually).

Contexts First, we decide which information we think is useful and important for prediction. For instance, suppose we want to take into account (in addition to the current or central word) the previous word, the next word and the previous tag.

Suppose that each sentence $(w_1, ..., w_N)$ is of size N, which varies from sentence to sentence. A priori, a context x_i (for the given sentence $w_1, ..., w_N$) is a function of $w_1, ..., w_N$ and $t_1, ..., t_{i-1}$ and the current position i. So, instead of having the tagged sentence $[(\mathbf{w_1}; \mathbf{t}_1), ..., (w_N; \mathbf{t}_N)]$ we'll have now:

$$\begin{array}{c}[(\_, \mathbf{w_1}, w_2, \_; \quad \mathbf{t_1}), ..., (w_{i-1}, \mathbf{w_i}, w_{i+1}, t_{i-1}; \quad \mathbf{t_i}), ..., (w_{N-1}, \mathbf{w_N}, \_, t_{N-1}; \quad \mathbf{t_N})] \\ = [(\mathbf{x_1}; \quad \mathbf{t}_1), ... , (\mathbf{x_i}; \quad \mathbf{t}_i), ... , (\mathbf{x_N}; \quad \mathbf{t}_N)]\end{array}$$

we have then built the corresponding N contexts $(x_1, ..., x_N)$ such that
$\mathbf{x_1} = (\_, \mathbf{w_1}, w_2, \_)$; $\mathbf{x_2} = (w_1, \mathbf{w_2}, w_3, t_1)$; $\mathbf{x_i} = (w_{i-1}, \mathbf{w_i}, w_{i+1}, t_{i-1})$; $\mathbf{x_N} = (w_{N-1}, \mathbf{w_N}, \_, t_{N-1})$

The bold words and tags represent the central word and its tag. These operations above are performed on each sentence in the corpus. Thus, the learning sample becomes $D^{MaxEnt} = \{(x_1, t_1), ..., (x_n, t_n)\}$ with n (of the same length as before) and represents the size of this learning data.

Features

Features are binary functions that encode context elements x useful for predicting the t tag (or y class). A feature f_j detects the co-occurrence of a certain prediction t with a certain context x.

$$f \ : \ \mathcal{X} \times \mathcal{Y} \to \{0; 1\} \text{ such as } f_{g,t}(x, t') = \begin{cases} 1 & \text{if g(x) = condition and t'=t} \\ 0 & \text{else} \end{cases}$$

where $g(x)$ is the projection of x on a space with a smaller dimension than x. That is, g(x) represents one (or more) coordinates of x. For instance, two features were selected: one that examines pairs (w_i ends with "able", t=adj) and (w_i ends with "able", t=NN). And based on the database, approximately 98% of words ending in "able" were adjectives ('admirable', 'applicable', 'available', etc.), while the remaining 2% were nouns ('table', 'cable', etc.).

Examples

The initial database can be represented by an $n \times 2$ matrix M that can be expressed as follows

$$M = \begin{pmatrix} w_1 & t_1 \\ w_2 & t_2 \\ \vdots & \vdots \\ w_i & t_i \\ \vdots & \vdots \\ w_{n-1} & t_{n-1} \\ w_n & t_n \end{pmatrix}.$$

The matrix M is then transformed to another huge binary matrix $M' \in \mathscr{M}_{n,q}(\{0;1\}) = \{0;1\}^{n\times q}$. M' might look like as shown below:

$$M' = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 1 & \cdots \\ 0 & 1 & & 0 & 1 & 0 & \cdots & 0 & 0 & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \\ 0 & 0 & & 1 & & & \vdots & & \vdots & \\ \vdots & \vdots & & \vdots & \vdots & & & \vdots & \vdots & \\ 0 & 1 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 & \cdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 & 0 & \cdots \end{pmatrix}$$

$$\begin{bmatrix} & \mathbf{w_i}{=}w_1 & {=}w_2 & \cdots & \mathbf{w_i}{=}w_k & \mathbf{w_{i-1}}{=}w_1 & {=}w_2 & \cdots & \mathbf{w_{i+1}}{=}w_1 & {=}w_2 & \cdots \\ t_1 & 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 1 & \cdots \\ t_2 & 0 & 1 & & 0 & 1 & 0 & \cdots & 0 & 0 & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & & \vdots & \vdots \\ t_k & 0 & 0 & & 1 & & & \vdots & & & \vdots \\ \vdots & \vdots & \vdots & & \vdots & \vdots & & & \vdots & \vdots & \\ t_{N-1} & 0 & 1 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 & \cdots \\ t_N & 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 & 0 & \cdots \end{bmatrix}$$

where q denotes the total number of features. It means that each column represents a feature. w_i means that we're looking to the current or central word, w_{i+1} and w_{i-1} refers respectively to the following and the preceding words. Also we stopped at $\mathbf{w_i} = w_k$ to say that there is k unique word among the n examples. (For the moment there is no assumption about the order of k). This enables us to directly calculate the conditional probability (without going through the joint probability) cf. (2.2) as follows. Let $w_1, ..., w_N$ be a sentence of length N from the D^{MaxEnt}. Now we can write the conditional probability (2.2) in a new form

$$p(t_1, ..., t_N \mid w_1, ..., w_N) = \prod_{i=1}^{N} p(t_i \mid x_i), \tag{2.3}$$

where $x_1, ..., x_N$ denotes the corresponding contexts sequence to the given sentence $w_1, ..., w_N$. Writing the likelihood or the probability in this way is going to be very important for what follows, one of the crucial points to keep in mind. Because in the same way that the x_i vectors are supposed to represent enough information from the sentence to predict the right tag for the word w_i, one could build another classifier that extracts information from the text and predicts its class in a different new problem.

2.2 Maximum Entropy

2.2.1 Background and links with linguistics and NLP

The concept of the principle of maximum entropy has a long history that has touched several fields such as mathematics, physics or economics, econometrics (Golan et al. (2007) [77], Golan (2008) et al. [79]) and computational linguistics. Basically, the idea is that for a given set of training examples, the Maximum Entropy principle say MaxEnt, enables to find a distribution which, satisfies the input constraints and maximizes the uncertainty. Certain researchers (Berger et al. (1996) [12]) consider Laplace as the father of maximum entropy, having enunciated the underlying theme more than two hundred years ago in his *Principle of Insufficient Reason*. And according to a more recent pioneer Edwin T Jaynes (1957) [95], he stated in [96] that "...the principle of MaxEnt may be considered as an

extension of Laplace's principle of insufficient reason, which stipulates that when one has no information to distinguish between the probability of two events, the best strategy is to consider them equally likely (see also Guiasu and Shenitzer (1985) [83]). Mathematically, the main property of MaxEnt is that no possibility is ignored as it assigns a positive weight to every situation that is not completely excluded by the given information (or input). Jaynes (1957) [95] finds that it's similar to an ergodic property. He states that "...in making inference on the basis of partial information we must use the probability distribution which has maximum entropy subject to whatever is known. This is the only unbiased assignment we can make".

Jaynes (1957) [95, 96] and Good (1963) [81] argued that the best probability model for the data is the one which maximizes entropy, over the set of probability distributions that are consistent with data and prior information. The MaxEnt principle, combined with some generalizations, is considered as a heuristic principle for generating null hypotheses. The main application is to a contingency table of an m-dimensional population, with marginal totals reduced to dimension $m-r$ ("constraints of r-th order"). The MaxEnt principle then leads to the null hypothesis, some cases of which have been treated by Bartlett and by Roy and Kastenbaum. It generalizes a conjecture due to Darroch et al. (1972) [57], who proved a kind of duality between maximum entropy and maximum likelihood on the one hand, and on the other hand described some relationships between maximum entropy, interactions, and Markov chains. By considering a random variable that is subject to a certain set of constraints, the MaxEnt principle allows to perform a test whose null hypothesis is such that the distribution is the one that maximizes entropy, under these constraints, see Good (1963) [81].

Inferring a function from insufficient information is a common occurring issue in statistics (Grendar (2006) [82]) and applied mathematics (Vinod (1982) [202]) as econometrics (Golan et al. (2007) [77], Golan (2008) et al. [79]), called inverse problems. Typically, the example of signal or image reconstruction from the results of certain measurements, the attribution of a probability density or a mass function under certain moment constraints, or the expectations of certain functions applied to the data (underlying random variables). Generally, the practical solution to such problems is to select an element p of the feasible set $\mathcal{P}$ by a more or less ad-hoc rule, usually by minimizing some functional such as the L_2-norm or negative entropy. If some function is specified as a "prior guess" or a given default distribution p_0, it is natural to minimize a measure of distance from the latter $D(p, p_0)$ most often the L_2-distance, or, for probability density or mass functions, Kullback's I-divergence (also called information for discrimination or cross-entropy, see Csiszar (1996) [55]). In his paper [54], Imre Csiszar (1991) mentioned some of the various reasons that have been put forward to justify the importance and efficiency of I-divergence minimization (introduced into statistics by Kullback (1959) [129] [105] as the method of minimum discrimination information) and entropy maximization. As said above, the recent widespread applications of Maximum entropy have been pioneered to a large extent by Jaynes (1982) [97]. Csiszar (1991) ([53][54]) has argued that the conditional limit theorems of Van Campenhout and Cover (1981) [198] and Csiszar (1984) [52] suggest the interpretation that the minimum I-divergence "updating" of a prior probability distribution to respect some moment constraints is a limiting form of Bayesian updating (adopting an axiomatic approach in Csiszar (1991) [54]).

2.2.2 Mathematical Formalisation of the MaxEnt principle

As considered in Csiszar (1996) [55], MaxEnt is a method to infer a measure $p(z)$ defined on a given set $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ under some constraint $\mathcal{P}$ (that specifies the only feasible set $\mathcal{P}$ of such functions). These available constraints can encode prior knowledge or some external information. Typically but not always, the feasible set is determined by linear constraints on p, i.e.,

$$\mathcal{P} = \left\{ p : \int f_k(z) p(z) l(dz) = \mu_k, k = 1, ...q \right\}$$

for some given funtions f_k and constants $\mu_k, k = 1, .., q$. l denotes a given σ-finite measure most of the time Lebesgue measure. The function p to be inferred is often a probability density or mass function.

Then, when the only available information is $p \in \mathcal{P}$, the MaxEnt solution to the problem is that $p^* \in \mathcal{P}$ maximizing the following (Shannon) entropy

$$H(p) = -\int p(z) \log p(z) l(dz).$$

If, in addition to the availability of p, we have access to a given default distribution $p_0 \in \mathcal{P}$, then the MaxEnt solution is $p^* \in \mathcal{P}$ that minimizes the information divergence (it is not a distance because it is not symmetric) from p_0 defined as follows

$$D(p, p_0) = \int \left[p(z) \log \frac{p(z)}{p_0(z)} - p(z) + p_0(z) \right] l(dz).$$

The term $-p(z) + p_0(z)$ cancels when we deal with densities. And

$$H(p) = H(p_0) - D(p, p_0)$$

for every probability density p_0. This divergence has a variety of other names and notations such as Kullback-Leibler, information number, relative entropy, information gain, etc. It is a non-symmetric measure of distance of p from p_0, i.e. $D(p, p_0) \geq 0$, with equality iff $p = p_0$ (interpreted as $p(z) = p_0(z)$ for l-almost all z). The function p^* minimizing $D(p, p_0)$ subject to $p \in \mathcal{P}$ is called the I-projection of p onto $\mathcal{P}$.

It is important to notice that a MaxEnt solution as defined above does not always exist. However, if a solution exists, it is unique, provided that the feasible set $\mathcal{P}$ is convex (see Csiszar (1996) [55]). Actually, $H(p)$ measures the average amount of information provided by the outcome of a random drawing governed by p , which may also be interpreted as a measure of uncertainty about that outcome before observing it, or of the amount of randomness represented by p. $D(p, p_0)$ is an information theoretic distance of p from p_0. It measures how less informed one is about the outcome of a random drawing one who believes this drawing is governed by p_0 than one who knows the true p. Alternatively, it is a measure of information gained when learning that the true distribution is p rather than p_0.

Using the dual form as in Borwein and Lewis (1992) [23], it is possible to show that the solution has the following exponential form, where λ_i's are the Kuhn & Tucker coefficients in the optimization program with each λ_i corresponding to a constraint μ_i,

$$p(z) = \frac{\exp\left(\sum_{k=1,..,q} \lambda_k \left(f_k(z) - \mu_k\right)\right)}{\int \exp\left(\sum_{k=1,..,q} \lambda_k \left(f_k(u) - \mu_k\right)\right) l(du)} = \frac{\exp\left(\sum_{k=1,..,q} \lambda_k f_k(z)\right)}{\int \exp\left(\sum_{k=1,..,q} \lambda_k f_k(u)\right) l(du)}.$$

If we consider the POS tagging case, by putting $z = (x, t)$ we can rewrite

$$p(z) = p(x, t) = p(t|x)p(x) = \frac{\exp\left(\sum_{k=1,..,q} \lambda_k f_k(x, t)\right)}{\int \exp\left(\sum_{k=1,..,q} \lambda_k f_k(u)\right) l(du)},$$

leading in the discrete case to

$$p(t|x) = \frac{\exp\left(\sum_{k=1,..,q} \lambda_k f_k(x, t)\right)}{\int \exp\left(\sum_{k=1,..,q} \lambda_k f_k(x, t')\right) l(dt')}.$$

In practice, we estimate $p(x)$ first using data, that's why it is called the a priori probability, then we plug it into these last expressions to get $p(t|x)$ called the a posteriori probability.

A simple illustrating example in NLP

The maximum entropy method is used for instance for the problem of translation from English to French as shown by Adam L. Berger et al. (1996) [12]. To illustrate in a simple way the main principle, we borrow the following example and explain why it generates exponential models. In their example, they focus on the translation of the word "in" by the following five possibilities $\{dans,\ en,\ \grave{a},\ au\ cours\ de,\ pendant\}$. They start affirming that without any external information or prior knowledge, we only have

$$p(dans) + p(en) + p(\grave{a}) + p(au\ cours\ de) + p(pendant) = 1.$$

To simplify notations, let $p_i = p(w_i)$ for $w_i \in \{dans,\ en,\ \grave{a},\ au\ cours\ de,\ pendant\}$. Then, the model p would be such as

$$p = \underset{p\in\mathcal{P}_1}{\arg\max}\{H(p)\} \quad \text{s.t} \quad \mathcal{P}_1 = \left\{p = (p_1, ..., p_5) \mid \sum p_i = 1\right\}$$

where $H(p) = -\sum p_i \log p_i$. Thus, for $\lambda > 0$

$$\frac{\partial H(p)}{\partial p_i} = 0 \Leftrightarrow \frac{\partial}{\partial p_i}\left(-\sum p_i \log p_i + \lambda\left(\sum p_i - 1\right)\right) = 0$$
$$\Leftrightarrow\ p_1 = ... = p_5 = e^{\lambda-1} \Rightarrow \forall i = 1, ..., 5\ \ p_i = \frac{1}{5} \text{ and } \lambda = 1 - \log 5.$$

In other words, $p(w_i) = 1/5$ for $w_i \in \{dans,\ en,\ \grave{a},\ au\ cours\ de,\ pendant\}$. Then, after noticing that the words "dans" and "en" occurred 30% of the time, now we want the model to take into account the following additional constraint

$$p(dans) + p(en) = 3/10$$
$$p(dans) + p(en) + p(\grave{a}) + p(au\ cours\ de) + p(pendant) = 1.$$

Thus, among the large potential probability distributions that are consistent with these two constraints, they stated that a reasonable choice for p is the most uniform, that is the distribution which allocates its probability as evenly as possible, subject to the constraints. The optimization program becomes

$$p = \underset{p\in\mathcal{P}_2}{\arg\max}\{H(p)\} \quad \text{s.t} \quad \mathcal{P}_2 = \left\{p = (p_1, ..., p_5) \text{ under } \sum p_i = 1 \text{ and } p_1 + p_2 = \frac{3}{10}\right\}.$$

The same calculus as above leads to

$$p_1 = p_2 = e^{\lambda_1+\lambda_2-1} \text{ and } p_3 = p_4 = p_5 = e^{\lambda_1-1},$$

where $\lambda_1, \lambda_2 > 0$ represent the Kuhn & Tucker coefficients that saturates both constraints respectively. Since $p_1 + p_2 = 3/10$ then $p_1 = p_2 = 3/20$. Similarly $p_3 = p_4 = p_5 = 7/30$. In other words

$$p(dans) = p(en) = 3/20\ ;\ p(\grave{a}) = p(au\ cours\ de) = p(pendant) = 7/30,$$

Several works have been done in linguistics using maximum entropy methods in different projects in NLP and machine translation (see Charniak (2000) [40]), and even in particular fields such as automatic text simplification. Consider Adwait Ratnaparkhi (1998) [165] who demonstrated in his thesis that several important types of natural language ambiguities can be resolved with state-of-the-art accuracy using only a single statistical model based on the principle of maximum entropy. He proved in his thesis that the implementation of this single statistical modeling technique based on MaxEnt combined with and external knowledge information sources (even if this prior information is poor), suffices to achieve state-of-the-art performance in several important tasks to the natural language processing community such as sentence boundary detection, part-of-speech tagging, parsing, and text categorization discussed

in his thesis. Thus, Maximum entropy probability models offer a clean way to combine diverse pieces of extracted information from texts (using features) in order to estimate the probability of a certain class or category given a certain entry. In 1996, R. Rosenfeld [170] used a maximum entropy approach for adaptive statistical language modeling. In 2000, McCallum and Freitag [135] used a variant of maximum entropy combined with Markov models for information extraction and segmentation. During the same year, several works on POS-tagging based on MaxEnt have followed, such as the works of Toutanova (2000) [194] by enriching the knowledge sources used in a maximum entropy part-of-speech-tagger, and Toutanova (2003) [193]. Later in 2008, Ekbal and al. built a Bengali part-of-speech tagger Based on the Maximum Entropy principle (Ekbal et al. (2008) [66]). Afterward, in 2013, György Szarvas (2013) [186] worked on Lexical Substitution using MaxEnt methods combined with delexicalized features in supervised learning. Two important reasons to make the Maximum Entropy principle a very practical method for text processing or more generally for NLP : (1) MaxEnt is not sensitive to parameter settings and (2) MaxEnt handles correlated features well, which is crucial in NLP and text processing models where many features are highly correlated.

2.3 Generalized Empirical Likelihood

2.3.1 Theoretical Foundations

History

Empirical likelihood (EL) was proposed by Thomas and Grunkemeier (1975) [191] in order to obtain better confidence intervals involving the Kaplan-Meier estimator in survival analysis. Some extensions for the case of survey sampling have been considered by Hartley & Rao in their (1968) paper [90]. Based on the idea of Thomas and Grunkemeier, Owen (1988) [151] established a general framework of EL for nonparametric inference. During the years 1988 to 1990, Owen generalized Wilk's theorem (1938) [205], stating that $-2\log(R)$ has asymptotically a χ^2 distribution, for a nonparametric framework (where R represents a likelihood ratio) (see Owen (1988) [151], Owen et al. (1990) [150]).

A possible interpretation of the empirical log-likelihood ratio is to consider it as the minimization of the Kullback divergence, say I_K, between the empirical distribution of the data $\mathbb{P}_n$ and a measure (or a probability measure) $\mathbb{Q}$ dominated by $\mathbb{P}_n$, under linear or non-linear constraints imposed on $\mathbb{Q}$ by the model. The use of other pseudo-metrics instead of the Kullback divergence I_K has been suggested by Owen et al. (1990) [150] and many other authors. For instance, the choice of relative entropy has led to "Entropy econometrics" in the econometric field (see Golan et al. (1996) [78]). Related results may be found in the probabilistic literature about divergence or the method of entropy in mean (see Leonard (2001) [119, 120, 121], Gamboa and Gassiat (1996) [75]). Some generalizations of the empirical likelihood method have also been obtained by using Cressie-Read discrepancies. This has led to some econometric extensions known as "generalized empirical likelihood" (see Newey and Smith (2004) [147]), even if the "likelihood" properties and in particular the Bartlett-correctability in these cases are lost. Bertail et al. (2014) [15] have shown that Owen's original method in the case of the mean can be extended to any regular convex statistical divergence or φ^*-discrepancy (where φ^* is a regular convex function) under weak assumptions, for general Hadamard differentiable functionals (see Bertail et al. (2007) [17], Bertail et al. (2015) [16]). They call this method "empirical energy minimizers" by analogy to the theoretical probabilistic literature on the subject (see Leonard (2001) [119, 120, 121] and the references therein).

The generalized empirical method is closely related to the maximum entropy method used in POS tagging via the notion of Dual likelihood introduced by Mykland (1995) [146]. Indeed the use of a specific divergence creates artificially a dual likelihood which is in the case of the entropy precisely the likelihood that is used in the MaxEnt method described before. This suggests several extensions, first to create new likelihoods, second to propose some procedures to take into account the large dimension aspects of the problem in text analysis.

A brief overview

Let $Z_1, ..., Z_n$ be independent identically distributed variables following $\rightsquigarrow \mathbb{P} \in \mathcal{P}$ (where $\mathcal{P}$ is now a convex set of probability). Z_i taking values on a space $\mathcal{Z}$ defined on $(\Omega, A, \mathbb{P}_\Omega)$. We are interested in constructing a confidence region for the functional parameter $\theta = \mathbf{T}(\mathbb{P})$ defined on ζ, taking values in $\mathbb{R}^q$. In the following, we define $\mathbb{P}_n$ the empirical probability measure as follows

$$\mathbb{P}_n = \frac{1}{n}\sum_1^n \delta_{Z_i}.$$

Owen (1990) [150] has shown that $\mathbb{P}_n$ is the NPMLE of $\mathbb{P}$ (Non Parametric Maximum Likelihood Estimate). Thus the NPMLE of $\mathbf{T}(\mathbb{P})$ is then its emprical counterpart $\hat{\theta}_n = \mathbf{T}(\mathbb{P}_n)$, called a statistical functional. Many statisticians (since Von Mises, see Serfling (1980) [175]) have been interested in deriving the asymptotic properties of $\hat{\theta}_n$ using differentiability assumptions on $\mathbf{T}$ via Taylor expansion (the delta method).

The empirical likelihood ratio evaluated at θ is defined by

$$R_{E,n}(\theta) = \sup_{\mathbb{Q}_n \in \mathcal{P}_n} \left\{ \prod_{i=1}^n \frac{d\mathbb{Q}_n}{d\mathbb{P}_n}(Z_i), \mathbf{T}(\mathbb{Q}_n) = \theta \right\},$$

where

$$\mathcal{P}_n = \left\{ \mathbb{Q}_n = \sum_{i=1}^n p_{i,n}\delta_{Z_i}, p_{i,n} \geq 0, \sum_{i=1}^n p_{i,n} = 1 \right\}$$

.

The log-likelihood ratio is thus

$$\log(R_{E,n}(\theta)) = \sup_{(p_{i,n})_{i\leq n}} \left\{ \sum_{i=1}^n \log\left(\frac{p_{i,n}}{\frac{1}{n}}\right), \mathbf{T}\left(\sum_{i=1}^n p_{i,n}\delta_{Z_i}\right) = \theta, \sum_{i=1}^n p_{i,n} = 1 \right\}.$$

A better way to see this problem from a probabilistic point of view is to consider the formula above as the minimization of the Kullback distance I_K between $\mathbb{Q}_n$ and $\mathbb{P}_n$, where

$$I_K(\mathbb{Q}, \mathbb{P}) = \begin{cases} -\int \log(\frac{d\mathbb{Q}}{d\mathbb{P}}) d\mathbb{P} & \text{if } \mathbb{Q} \ll \mathbb{P} \\ +\infty & \text{else} \end{cases}$$

under the two constraints, on the parameter and the probabilities p_i.

For instance, $\mathbf{T}(\mathbb{P})$ may be the unique solution of some estimating equations $E_{\mathbb{P}} f(Z, \mathbf{T}(\mathbb{P})) = 0$ (see Qin and Lawless (1994) [158]) where for each fixed parameter $\mathbf{T}(\mathbb{P})$, f is a measurable function defined from $\mathcal{Z}$ to $\mathbb{R}^q$, $q \geq 1$. These equations will also include marginal constraints (that is constraints independent of the parameter $\mathbf{T}(\mathbb{P})$ incorporating some knowledge of the data: see an application on large datasets of this kind of idea in Crepet et al. (2009) [51]). In this case, the constraint becomes $E_{\mathbb{Q}_n} f(Z, \theta) = 0 = \sum p_{i,n} f(Z_i, \theta)$ and the empirical likelihood boils down to the convex maximization program

$$R_{E,n}(\theta) = \sup_{p_{i,n}, i=1,...,n} \left\{ \begin{array}{c} \frac{\Pi_{i=1}^n p_{i,n}}{1/n^n} \text{ under } \sum_{i=1}^n p_{i,n} f(Z_i, \theta) = 0 \\ \sum_{i=1}^n p_{i,n} = 1,\ p_{i,n} \geq 0 \end{array} \right\}.$$

Some standard results in convex optimization theory give conditions for this problem to have a solution and also allow to obtain a dual representation of this problem. This is precisely the dual representation that generates the MaxEnt model used in NLP as we will explain below.

Since the case of estimating equation $E_P f(Z, \mathbf{T}(P)) = 0$ is mostly relevant in our framework we will essentially consider this case from now on.

2.3.2 Generalized empirical likelihood and MaxEnt models

A general view of empirical likelihood

Consider a measured space $(\mathcal{Z}, \mathcal{A}, \mathcal{M})$ where $\mathcal{M}$ is a space of signed measures. Working on a space of signed measures will be essential for applications to ensure the existence of solutions of the original optimization program. Let f be a measurable function defined from $\mathcal{Z}$ to $\mathbb{R}^q$, $q \geq 1$. For any measure $m \in \mathcal{M}$, we write $mf = \int f dm$. In the following, we consider φ, a convex function whose support $d(\varphi)$, defined as $\{x \in \mathbb{R},\ \varphi(x) < \infty\}$, is assumed to be non-void (that is φ is proper). We denote respectively $\inf d(\varphi)$ and $\sup d(\varphi)$, the extremes of this support. For every convex function φ, its convex dual or Fenchel-Legendre transform is given by

$$\varphi^*(y) = \sup_{x \in \mathbb{R}} \{xy - \varphi(x)\},\ \forall\ y \in \mathbb{R}.$$

Recall that φ^* is then a semi-continuous inferior (s.c.i.) convex function. We define by $\varphi^{(i)}$ the derivative of order i of φ when it exists. From now on, we will assume the following assumptions for the function φ. We adopt the same notations as in Bertail et al. (2007) [17].

H1 φ is strictly convex and $d(\varphi)$ contains a neighborhood of 0 ;

H2 φ is twice differentiable on a neighborhood of 0 ;

H3 (renormalization) $\varphi(0) = 0$ and $\varphi^{(1)}(0) = 0$, $\varphi^{(2)}(0) > 0$, which implies that φ has an unique minimum at zero ;

H4 φ is differentiable on $d(\varphi)$, that is to say, differentiable on $int\{d(\varphi)\}$, with right and left limits on the respective endpoints of the support of $d(\varphi)$, where $int\{.\}$ is the topological interior.

H5 φ is twice differentiable on $d(\varphi) \cap \mathbb{R}^+$ and, on this domain, the second order derivative of φ is bounded from below by a constant $\varphi_{\min} > 0$.

Let φ satisfies the hypotheses **H1, H2, H3**. Then, the Fenchel dual transform φ^* of φ also satisfies these hypotheses. The φ^*-discrepancy I_{φ^*} between $\mathbb{Q}$ and $\mathbb{P}$, where $\mathbb{Q}$ is a signed measure and $\mathbb{P}$ a positive measure, is defined as follows

$$I_{\varphi^*}(\mathbb{Q}, \mathbb{P}) = \begin{cases} \int_{\mathcal{X}} \varphi^* \left(\frac{d\mathbb{Q}}{d\mathbb{P}} - 1 \right) d\mathbb{P} & \text{if } \mathbb{Q} \ll \mathbb{P}, \\ +\infty & \text{else.} \end{cases} \tag{2.4}$$

For details on φ^*-discrepancies or divergences and some historical comments, see Liese and Vajda (1987) [123], Leonard (2001) [119, 120, 121]. It is easy to check that Cressie-Read discrepancies fulfill these assumptions. Indeed, a Cressie-Read discrepancy can be seen as a φ^*-discrepancy, with φ^* given by

$$\varphi^*_\kappa(x) = \frac{(1+x)^\kappa - \kappa x - 1}{\kappa(\kappa - 1)},\ \varphi_\kappa(x) = \frac{[(\kappa - 1)x + 1]^{\frac{\kappa}{\kappa - 1}} - \kappa x - 1}{\kappa}$$

for some $\kappa \in \mathbb{R}$. This family contains all the usual discrepancies, such as relative entropy ($\kappa \to 1$), Hellinger distance ($\kappa = 1/2$), the χ^2 ($\kappa = 2$) and the Kullback distance ($\kappa \to 0$) (see the annex for more details).

For us, the main interest of φ^*-discrepancies relies on the following duality representation, which follows from the results of Borwein and Lewis (1991) [22] on convex functional integrals (see also Rockafellar (1968) [168]). The following theorem in Bertail et al. (2007) [17] is a simplified version of the results by Borwein and Lewis (1991) [22]. We also refer to Keziou (2003) [102] and Broniatowski and Keziou (2006) [30] for other dual representations and a very precise study of the topological aspects of the problem.

Theorem 2.3.1. *Let $\mathbb{P} \in \mathcal{M}$ be a probability measure with a finite support and f be a measurable function on $(\mathcal{Z}, \mathcal{A}, \mathcal{M})$. Let φ be a convex function satisfying assumptions **H1-H3**. If the following qualification constraint holds,*

$$Qual(\mathbb{P}): \begin{cases} \exists \mathbf{T} \in \mathcal{M}, \mathbf{T}f = \mu_0 \text{ and} \\ \inf d(\varphi^*) \ < \ \inf_{\mathcal{Z}} \frac{d\mathbf{T}}{d\mathbb{P}} \ \leq \ \sup_{\mathcal{Z}} \frac{d\mathbf{T}}{d\mathbb{P}} \ < \ \sup d(\varphi^*) \ \ \mathbb{P}-a.s., \end{cases}$$

then, we have the dual equality:

$$\inf_{\mathbb{Q} \in \mathcal{M}} \{I_{\varphi^*}(\mathbb{Q}, \mathbb{P}) | \ (\mathbb{Q} - \mathbb{P})f = \mu_0\} = \sup_{\lambda \in \mathbb{R}^q} \left\{ \lambda' \mu_0 - \int_{\mathcal{Z}} \varphi(\lambda' f) d\mathbb{P} \right\}. \tag{2.5}$$

*If φ satisfies **H4**, then the supremum on the right hand side of (2.5) is achieved at a point λ^* and the infimum on the left hand side at $\mathbb{Q}^*$ is given by*

$$\mathbb{Q}^* = (1 + \varphi^{(1)}(\lambda^{*\prime} f))\mathbb{P}.$$

The same kind of results also holds when the number of constraints goes to infinity or even is infinite, see Leonard (2001) [119, 120, 121] and Broniatowki and Keziou (2012) [31] for some applications to a continuum of moment constraints.

Empirical optimization of φ^*-discrepancies

Let $Z_1, ...Z_n$ be i.i.d. r.v.'s defined on $\mathcal{Z}$ with common probability measure $\mathbb{P} \in \mathcal{M}$. We will here consider that the parameter of interest $\theta \in \mathbb{R}^q$ is the solution of some M-estimation problem $\mathbb{E}_{\mathbb{P}} f(Z, \theta) = 0$, where f is now a regular differentiable function from $\mathcal{Z} \times \mathbb{R}^q \to \mathbb{R}^r$. For simplicity, we now assume that f takes its value in $\mathbb{R}^q$, that is $r = q$, and that there is no over-identification problem. The over-identified case can be treated similarly by first reducing the problem to the strictly identified case (see Qin and Lawless (1994) [158]).

For a given φ, we define, by analogy to the empirical likelihood problem, the quantity

$$\beta_n(\theta) = n \inf_{\mathbb{Q} \in \mathcal{P}_n} \{I_{\varphi^*}(\mathbb{Q}, \mathbb{P}_n)\} \quad \text{with} \ \ \mathcal{P}_n = \{\mathbb{Q} \in \mathcal{M} \mid \mathbb{Q} \ll \mathbb{P}_n, \ \mathbb{E}_{\mathbb{Q}} f(Z, \theta) = 0\}$$

We define the corresponding random confidence region

$$C_n(1-\alpha) = \{\theta \in \mathbb{R}^q \ | \exists \mathbb{Q} \ll \mathbb{P}_n \text{ with } \mathbb{E}_{\mathbb{Q}} f(Z, \theta) = 0 \text{ and } n I_{\varphi^*}(\mathbb{Q}, \mathbb{P}_n) \leq \eta(\alpha)\},$$

where $\eta(\alpha)$ is a quantity such that

$$\Pr(\theta \in C_n(1-\alpha)) = 1 - \alpha + o(1).$$

Define $\mathcal{M}_n = \{\mathbb{Q} \in \mathcal{M} \text{ with } \mathbb{Q} \ll \mathbb{P}_n\} = \{\mathbb{Q} = \sum_{i=1}^n p_{i,n} \, \delta_{Z_i}, \ (p_{i,n})_{1 \leq i \leq n} \in \mathbb{R}^n\}$. Considering this set of measures, instead of a set of probabilities, can be partially explained by Theorem 2.3.1.

The underlying idea of empirical likelihood and its extensions is actually a plug-in rule. Consider the functional defined by

$$M(\mathbb{P}, \theta) = \inf_{\{\mathbb{Q} \in \mathcal{M}, \ \mathbb{Q} \ll \mathbb{P}, \ \mathbb{E}_{\mathbb{Q}} g(Z, \theta) = 0\}} I_{\varphi^*}(\mathbb{Q}, \mathbb{P})$$

that is the minimization of a contrast under the constraints imposed by the model. This can be seen as a projection of $\mathbb{P}$ on the model of interest for the given pseudo-metric I_{φ^*}. If the model is true at $\mathbb{P}$, that is, if $\mathbb{E}_{\mathbb{P}} f(Z, \theta) = 0$, then clearly $M(\mathbb{P}, \theta) = 0$. A natural estimator of $M(\mathbb{P}, \theta)$ for fixed θ is given by the plug-in estimator $M(\mathbb{P}_n, \theta)$, which is $\beta_n(\theta)/n$. This estimator can then be used to test $M(\mathbb{P}, \theta) = 0$ or, in a dual approach, to build confidence region for θ by inverting the test.

For $\mathbb{Q}$ in $\mathcal{M}_n$, the constraints can be rewritten as $(\mathbb{Q}-\mathbb{P}_n)f(.,\theta) = -\mathbb{P}_n f(.,\theta)$. Using Theorem 2.3.1 or the results of Broniatowki and Keziou (2006) [30], we get the dual representation

$$\begin{aligned}\beta_n(\theta) &:= n \inf_{\mathbb{Q}\in\mathcal{M}_n} \{I_{\varphi^*}(\mathbb{Q},\mathbb{P}_n),\ (\mathbb{Q}-\mathbb{P}_n)f(.,\theta) = -\mathbb{P}_n f(.,\theta)\} \\ &= n \sup_{\lambda\in\mathbb{R}^q} \mathbb{P}_n\left(-\lambda' f(.,\theta) - \varphi(\lambda' f(.,\theta))\right). \end{aligned} \tag{2.6}$$

Notice that $-x-\varphi(x)$ is a strictly concave function and that the function $\lambda \to \lambda' f$ is also concave. The parameter λ can be simply interpreted as the Kuhn & Tucker coefficient associated with the original optimization problem. From this representation of $\beta_n(\theta)$, we can now derive the usual properties of the empirical likelihood and its generalization. In the following, we will also use the notations

$$\overline{f}_n = \mathbb{P}_n f(.,\theta) = \frac{1}{n}\sum_{i=1}^n f(Z_i,\theta)\ ;\ S_n^2 = \frac{1}{n}\sum_{i=1}^n f(Z_i,\theta)f(Z_i,\theta)' \text{ and } S_n^{-2} = (S_n^2)^{-1}.$$

then, under an empirical qualification constraint, $C_n(1-\alpha)$ is a convex asymptotic confidence region with

$$\begin{aligned}\lim_{n\to\infty} \Pr(\theta \notin C_n(1-\alpha)) &= \lim_{n\to\infty} \Pr(\beta_n(\theta) \geq \eta) \\ &= \lim_{n\to\infty} \Pr\left(n\overline{f}_n' S_n^{-2}\overline{f}_n \geq \chi_q^2(1-\alpha)\right) \\ &= 1-\alpha.\end{aligned}$$

where $\chi_q^2(1-\alpha)$ represents the quantile of the order $1-\alpha$ of a χ^2 distribution with q degress of freedom.

Empirical likelihood and the Kullback discrepancy In the particular case $\varphi_0(x) = -x-\log(1-x)$ and $\varphi_0^*(x) = x-\log(1+x)$ corresponding to the Kullback divergence for measures

$$K(\mathbb{Q},\mathbb{P}) = -\int \log(\frac{d\mathbb{Q}}{d\mathbb{P}})d\mathbb{P} + \int (d\mathbb{Q}-d\mathbb{P}),$$

the dual program obtained in (2.6) becomes, for the admissible θ,

$$\beta_n(\theta) = \sup_{\lambda\in\mathbb{R}^q}\left(\sum_{i=1}^n \log(1-\lambda' f(Z_i,\theta))\right).$$

As a parametric likelihood indexed by λ, it is easy to show that $2\beta_n(\theta)$ is asymptotically $\chi^2(q)$ when $n\to\infty$, if the variance of $f(Z,\theta)$ is definite. It is also Bartlett-correctable since it is a likelihood in λ in its dual form (see Bertail (2006) [13]). For a general discrepancy, the dual form is not a likelihood and may not be Bartlett-correctable, see DiCiccio et al. (1991) [63].

Moreover, we necessarily have the $p_{i,n}'s > 0$, and the optimization program implies in this case that $\sum_{i=1}^n p_{i,n} = 1$, that is the solution is a probability so that the qualification constraint essentially means that 0 belongs to the convex hull of the $f(Z_i,\theta)$. This is in particular the reason which one may obtain very bad coverage probability for empirical likelihood.

GMM and χ^2 discrepancy The particular case of the χ^2 discrepancy corresponds to $\varphi_2(x) = \varphi_2^*(x) = \frac{x^2}{2}$. $\beta_n(\theta)$ can be explicitly calculated. Indeed, we get easily that the optimal value is $\lambda^* = -S_n^{-2}\overline{f}_n$. By Theorem 2.3.1, the minimum is attained at $\mathbb{Q}^* = \sum_{i=1}^n p_{i,n}\delta_{Z_i}$ with

$$p_{i,n} = \frac{1}{n}(1-\overline{f}_n' S_n^{-2} f(Z_i,\theta))$$

and

$$I_{\varphi_2^*}(\mathbb{Q}^*, \mathbb{P}_n) = \sum_{i=1}^{n} \frac{(np_{i,n} - 1)^2}{2n} = \frac{1}{2}\overline{f}'_n S_n^{-2} \overline{f}_n,$$

which is exactly a quadratic form of a self-normalized sum up to a factor n, which typically appears in the Generalized Method of Moments (GMM). In the third chapter of this thesis, we will study more precisely a penalized version of this quantity when the dimension q is very large.

Notice that, in opposition to the Kullback discrepancy, we may charge positively some regions outside of the convex hull of the points, yielding bigger (that is too conservative) confidence regions. Note that in this case, it is possible to get exact exponential bounds for this quantity as shown in Bertail et al. (2008) [14]. As a consequence even if $q << n$ eventually grows with n, it is still possible to get an automatic confidence region, by just relying on the internal optimization problem, without having to invert the empirical covariance matrix (which may be complicated) for big datasets with a lot of variables. We will later focus on what happens when the dimension in q is very large and of the same order as n.

Relative entropy: link to the MaxEnt problem

The particular case of the relative entropy corresponds to the convex function

$$\varphi^*(x) = (x+1)\log(1+x) - x$$

whose convex conjugate is given by $\varphi(x) = e^x - 1 - x$. In that case, the relative entropy defined for measures is given by

$$I_{\varphi^*}(\mathbb{Q}, \mathbb{P}) = \begin{cases} \int_{\mathcal{X}} \frac{d\mathbb{Q}}{d\mathbb{P}} \log(\frac{d\mathbb{Q}}{d\mathbb{P}}) d\mathbb{P} - \int (d\mathbb{Q} - d\mathbb{P}) & \text{if } \mathbb{Q} \ll \mathbb{P} \\ +\infty & \text{else.} \end{cases}$$

the dual program obtained in (2.6) becomes, for the admissible θ

$$\beta_n(\theta) = \sup_{\lambda \in \mathbb{R}^q} \left(n - \sum_{i=1}^{n} \exp\left(\lambda' f(Z_i, \theta)\right) \right), \tag{2.7}$$

so that the optimal weights are given by

$$p_{i,n} = \frac{1}{n} \exp\left(\lambda^{*\prime} f(Z_i, \theta)\right),$$

where λ^* is the solution of 2.7 and is asymptotically equivalent to $-S_n^{-2}\overline{f}_n$ (just like in the GMM program). Notice that if in addition, we add the probability constraint as in the MaxEnt problem, we obtain the same empirical formulation as in the MaxEnt problem, that is, weights of the form

$$p_{i,n} = \frac{\exp\left(\lambda^{*\prime} f(Z_i, \theta)\right)}{\sum_{i=1}^{n} \exp\left(\lambda^{*\prime} f(Z_i, \theta)\right)}.$$

2.3.3 Penalizing the dual likelihood in large dimension

Some proposals in the large dimension case

The previous results and the asymptotic validity of generalized empirical likelihood essentially hold when q the number of constraints (equal for us to the dimension of the parameter θ) is fixed and small compared to n. In some recent work McKeague and von Keilegom (2009) [92] have studied the validity of empirical likelihood when q depends on n and such that $q << n^{1/3}$. They show that empirical likelihood still works: this can be explained by the fact that in that case, the empirical variance automatically

computed by the internal optimization program is still a convergent estimator of the true variance. However, as noticed by several authors, the method fails when the number of constraints tends to be too big, in particular when it is of the same size as n, see Lahiri and Mukhopadhyay (2012) [107] and Bartolucci (2007) [9].

Several propositions have emerged to treat large dimension problems with generalized empirical likelihood. We may classify them into three classes (or combinations of the three methods).

(i) **Enlarge-the-margin methods**: by this, we mean that instead of the original empirical likelihood problem, allow for some flexibility or some perturbations of the original constraints. This can be done either by adding one or several points to the data which do not have exactly the correct mean (see Chen et al. (2008) [42], Emerson and Owen (2009) [67]). Or this can be done by replacing the original constraints with some inequality constraints with respect to some norm $\|.\|_R$ defined by $\|x\|_R = x'Rx$, where R is possibly random allowing for some flexibility in the constraints. This leads to a relaxed empirical likelihood version

$$R_{E,n}^{pen}(\theta) = \sup_{(p_{i,n})_{i\leq n}} \left\{ \begin{array}{c} n^n \prod_{i=1}^n p_{i,n} \text{ under } \left\|\sum_{i=1}^n p_{i,n} f(Z_i,\theta)\right\|_R \leq \delta_n \\ \sum_{i=1}^n p_{i,n} = 1,\ p_{i,n} \geq 0 \end{array} \right\} \tag{2.8}$$

where δ_n is a margin to be calibrated (possibly depending on the data).

(ii) **Penalize the empirical likelihood** either on the primal form or the dual form. It is well known in the convex literature that program 2.8 may also be rewritten

$$log(R_{E,n}^{pen}(\theta)) = \sup_{p_{i,n},i=1,\ldots,n} \left\{ \begin{array}{c} \sum_{i=1}^n \log(p_{i,n}) - C_n(\delta_n)||\sum_{i=1}^n p_{i,n} f(Z_i,\theta)||_R \\ \sum_{i=1}^n p_{i,n} = 1,\ p_{i,n} \geq 0 \end{array} \right\}$$

which may be interpreted as a penalized version of the original program. Such penalizations have been studied in Bartolucci (2007) [9] and Lahiri and Mukhopadhyay (2012) [107] when $f(Z_i,\theta) = Z_i - \theta$, $Z_i = (Z_i^1, \ldots Z_i^q) \in \mathbb{R}^q$. The proposition of Bartolucci (2007) [9] corresponds to the choice $R = \hat{S}_n^{-2}$ and $\|x\|_R = x'\hat{S}_n^{-2}x$, $C_n(h) = n/2h^2$, and $\hat{S}_n^2$ is the sample covariance matrix

$$\hat{S}_n^2 = \frac{1}{n}\sum_{i=1}^n (Z_i - \overline{Z}_n)(Z_i - \overline{Z}_n)'.$$

Notice that this proposition may cause problems when q is bigger than n, since in that case, the sample covariance matrix is not full rank and thus not invertible. The proposition of Lahiri and Mukhopadhyay (2012) [107] in a more general dependent framework (the $(Z_i)_{i=1,\ldots,n}$ may be weak mixing or with long-range dependence) corresponds to

$$R = \text{diag}(\hat{\sigma}_j^{-2})_{j=1,\ldots,q},$$

with $C(h) = h$, where we use

$$\hat{\sigma}_j^2 = \frac{1}{n}\sum_{i=1}^n (Z_i^j - \bar{Z}_n^j)^2$$

the marginal empirical variances and Z_i^j and $\bar{Z}_n^j$ the jth component of Z_i and $\bar{Z}_n$ respectively. One purpose of Chapter 3 is further to explore the effect and automatic choice of penalization. Notice that for a very large dimension, the second proposal is expected to work better since $\hat{S}_n^2$ is singular if $q > n$.

Another proposition is to penalize the empirical likelihood in its dual form (see Mykland (1995) [146]) for an introduction to dual likelihood. A penalized version in the dual form has been recently studied by Otsu (2007) [149], and Chang et al. (2018) [39]. The most important results have been obtained by Shi (2016) [179] who proved that, for empirical likelihood with a correct penalization, the number of

constraints may be as large as $o(\exp(n^{1/3}))$. For generalized empirical likelihood, this corresponds to studying a penalized version of the dual program of the form

$$P_n(\theta,\lambda) = \mathbb{P}_n\left(- \lambda' f(.,\theta) - \varphi(\lambda' f(.,\theta))\right) - \frac{1}{2}||\lambda||_R^2,$$

and the corresponding optimization problem

$$\sup_{\lambda\in\mathbb{R}^q}(P_n(\theta,\lambda)),$$

which is clearly linked to the proposition of Bartolucci (2007) [9] and Lahiri and Mukhopadhyay (2012) [107] by duality consideration. We will not investigate here the relationship between the different dual formulations: however, this would be clearly of interest in particular when one uses the L_1 or the L_∞ norms or a combination of these norms with L_2 (elastic net) instead of a simple L_2 norm.

iii) **Choose another divergence** (on space of signed measure). Another proposition is to use a different criterion than the likelihood criterion, arising from the choice of measuring the distance between Q_n and P_n with the Kullback-Leibner divergence say $KL(Q,P) = -\int \log\left(\frac{dQ}{dP}\right) dP$. It is known for instance that the choice of $\chi^2(Q,P) = \int(\frac{dQ}{dP} - 1)^2 dP$ leads to the exact computation of the generalized empirical likelihood version provided that one works with signed measures Q_n dominated by P_n rather than probability measure (that is, one does not impose $p_{i,n} \geq 0$ and $\sum_{i=1}^n p_{i,n} = 1$). In this case, the maximization problem becomes

$$\begin{aligned} R_{\chi^2,n}(\theta) &= \sup_{p_{i,n},i=1,\ldots,n}\left\{\sum(\frac{p_{i,n}}{1/n} - 1)^2 \text{ under} \sum_{i=1}^n p_{i,n}(Z_i - \theta) = 0\right\}, \\ &= \frac{1}{2}(\overline{Z}_n - \theta)^{'} S_n^{2-}(\overline{Z}_n - \theta) \end{aligned}$$

where $S_n^2 = \frac{1}{n}\sum(Z_i - \theta)(Z_i - \theta)'$ and S_n^{2-} is its Moore-Penrose generalized inverse. For general constraints, the solution is close to the so-called GMM program.

Note that in the χ^2 case the dual problem (ii) when $R = \rho_n^{-1} I$ for some constant ρ_n, and with the choice of an L_2 penalization then the optimization program becomes

$$D = \sup_{\lambda\in\mathbb{R}^q} \mathbb{P}_n\left\{-\lambda^{'} f - (\lambda' f)^2/2 - \rho_n \lambda'\lambda\right\}$$

and the solution of this program is simply the regularized Hotelling statistics

$$\frac{1}{2} P_n f^{'}(P_n f f^{'} + \rho_n I)^{-1} P_n f$$

which is a regularized form of the T^2 Hotelling statistics (with no centering).

When the dimension $p << n$, Bertail et al. (2008) [14] have shown that one can choose $\rho_n = 0$ and can get some exact exponential bounds for this quantity. In Chapter 3, we investigate conditions to obtain exponential bounds in this large-dimension framework by choosing an adequate value for the penalty ρ_n. The GMM case when there is an infinite number (or a continuum) of constraints has been treated by several authors in the econometric literature, see for instance Carasco and Florens (2000) [34], using some Tikhonov regularization of the operator S_n^2.

2.4 A Penalized MaxEnt method: application to POS-tagging

In the following, we apply the ideas of generalized empirical likelihood developed in section 2.3 to the POS-tagging problem described in 2.2.

Let us remember that we have at our disposal a corpus $C = \{(w_i, t_i)\}_{i=1\ldots n}$. We transform C to a new dataset $D = \{(x_i, t_i)\}_{i=1\ldots n}$ where x_i's are the contexts of w_i's. Each x_i represents a new vector containing the current word w_i and the surrounding information (including words, punctuation, affixes of the current word, etc.). We intend to estimate $p(t_i \mid x_i)$ using the generalized empirical likelihood framework (Bertail (2006) [13]) equipped with relative entropy divergence. For two probability distributions $\mathbf{p} = (p_1, ..., p_n)$ and $\mathbf{q} = (q_1, ..., q_n)$, recall that D_K is given by :

$$D_K(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n} p_i \log\left(\frac{p_i}{q_i}\right)$$

Now we assume that we observe n i.i.d r.v's $Z_i = (x_i, t_i)$ having some distribution P. Notice that by a slight abuse of notation for f that in this framework the multidimensional function $f(Z, \theta)$ is in fact given by the vector $f(t, x) - \mu = (f_k(t, x) - \mu_k, k = 1, \ldots, q)$, with $\theta = \mu = \{\mu_k, k = 1, \ldots, q\}$

2.4.1 Relative entropy and MaxEnt problem

It is easy to check that the dual of the initial minimization problem is given by

$$\sup_{\lambda \in \mathbb{R}^q} \left\{ 1 - \frac{1}{n} \sum_{i=1}^{n} e^{\lambda'(f(x_i,t_i)-\mu)} \right\} \tag{2.9}$$

We see at once that

$$\forall \lambda \in \mathbb{R}^q, \lambda' \left(\sum_{i=1}^{n} p(x_i) p_i \left(f(x_i, t_i) - \mu \right) \right) = 0$$

which is clear from constraints.

An easy reformulation makes it obvious that

$$p(x_i)\, p(t_i|x_i) = \frac{1}{n} e^{\lambda'(f(x_i,t_i)-\mu)}$$

Since $\forall x_i$, $\sum_{k=1}^{T} p(t_k \mid x_i) = 1$, it follows that

$$p_i = p(t_i \mid x_i) = \frac{e^{\lambda'\mu} \cdot e^{-\lambda' f(x_i,t_i)}}{e^{\lambda'\mu} \cdot \sum_{k=1}^{T} e^{-\lambda' f(x_i,t_k)}}.$$

This is equivalent and justify the use of formula (2.9) in the MaxEnt program.

Finally, we obtain

$$p(t_i \mid x_i) = \frac{e^{-\lambda' f(x_i,t_i)}}{\sum_{k=1}^{T} e^{-\lambda' f(x_i,t_k)}} = \frac{e^{-\sum_{j=1}^{q} \lambda_j \cdot f_j(x_i,t_i)}}{\sum_{k=1}^{T} e^{-\sum_{j=1}^{q} \lambda_j \cdot f_j(x_i,t_k)}} \tag{2.10}$$

This proves that minimizing the Relative entropy divergence between the desired distribution and the multinomial distribution gives the same solution as the one obtained when maximizing the likelihood of a log-linear model.

Moreover, we get the predictive probability of t_i given x using the estimate

$$p\left(t_i|x\right) = \frac{e^{-\lambda' f(x,t_i)}}{\sum\limits_{t_i \in \mathcal{T}} e^{-\lambda' f(x,t_i)}}$$

We know from the duality results exposed before that the optimal value of λ is asymptotically given by $\lambda^* = -S_n^{-2}(\bar{f}_n - \mu)$. Unfortunately, this quantity depends on μ. It may be estimated in two different ways according to the context we are interested in.

- Method 1 : Either estimate μ by estimating the log-linear model considered in the MaxEnt method or by using the method proposed in Quin and Lawless(1994) [158] that is, find the value of $\hat{\mu}$ which realizes

$$\inf_{\mu \in \mathbb{R}^q} \sup_{\lambda \in \mathbb{R}^q} \left\{ 1 - \frac{1}{n} \sum_{i=1}^{n} e^{\lambda'(f(x_i,t_i)-\mu)} \right\} \tag{2.11}$$

  this implies that asymptotically we have $\hat{\lambda}^* = -\hat{S}_n^{-2}(\bar{f}_n - \hat{\mu})$ with

$$\hat{S}_n^2 = \frac{1}{n} \sum_{i=1}^{n} \left(f(x,t_i) - \hat{\mu}\right) \left(f(x,t_i) - \hat{\mu}\right)'.$$

  This yields an asymptotic expression for the conditional probability given by

$$\hat{p}\left(t_i|x\right) = \frac{e^{-(\bar{f}_n - \hat{\mu})' \hat{S}_n^{-2} f(x,t_i)}}{\sum\limits_{t_k \in \mathcal{T}} e^{-(\bar{f}_n - \hat{\mu})' \hat{S}_n^{-2} f(x,t_k)}}.$$

  The advantage of this expression is that it does not require the computational optimization used for the log-linear model proposed in the MaxEnt method.

- Method 2 : In some situations, for a given context (complex text or simplified text), we can observe another corpus and compute an estimator $\tilde{\mu}$ of μ. In that case, we can use directly this estimator to get the predictive probability

$$\tilde{p}\left(t_i|x\right) = \frac{e^{-(\bar{f}_n - \tilde{\mu})' \tilde{S}_n^{-2} f(x,t_i)}}{\sum\limits_{t_k \in \mathcal{T}} e^{-(\bar{f}_n - \tilde{\mu})' \tilde{S}_n^{-2} f(x,t_k)}}$$

  with

$$\tilde{S}_n^2 = \frac{1}{n} \sum_{i}^{n} \left(f(x,t_i) - \tilde{\mu}\right) \left(f(x,t_i) - \tilde{\mu}\right)'.$$

Since in POS-tagging the number of tags is relatively limited such computation is almost immediate.

2.4.2 The penalized version of MaxEnt

A natural question is to propose a method adapted to large dimension constraints in the framework of NLP. For this consider the problem of generalized empirical likelihood with an L_2 penalization.

The penalized empirical divergence in the relative entropy case is

$$P_n(\mu,\lambda) = \mathbb{P}_n \left(-\lambda' \left(f\left(x,t\right) - \mu\right) - \varphi(\lambda' \left(f\left(x,t\right) - \mu\right)) \right) - \frac{1}{2} ||\lambda||_R^2.$$

and becomes

$$P_n(\mu,\lambda) = 1 + \frac{1}{n}\sum_{i=1}^{n}\left(-\exp(\lambda'\left(f\left(x_i,t_i\right)-\mu\right)) - \frac{1}{2}||\lambda||_R^2\right).$$

Notice that when $R = \rho_n I$, then this quantity becomes asymptotically for λ close to 0 (as expected),

$$P_n(\mu,\lambda) \approx \frac{1}{n}\sum_{i=1}^{n}\left(-\lambda'\left(f\left(x_i,t_i\right)-\mu\right)) - \frac{1}{2}\lambda'(S_n^2+\rho_n I)\lambda\right)$$

whose maximum is attained at

$$\lambda_n^* = -(S_n^2+\rho_n I)^{-1}\frac{1}{n}\sum_{i=1}^{n}\left(f\left(x_i,t_i\right)-\mu\right) = -(S_n^2+\rho_n I)^{-1}P_n(f-\mu)$$

yielding the value at the optimum

$$\frac{1}{2}P_n(f-\mu)^{'}(P_n(f-\mu)(f-\mu)^{'}+\rho_n I)^{-1}P_n(f-\mu),$$

as in the χ^2 case (for which the expression was exact). Chapter 3 will specifically focus on obtaining a precise exponential control of such quantity.

In the penalized case we see that the optimal weights depend on μ and are given by

$$\hat{p}\left(t_i|x\right) = \frac{e^{-(\bar{f}_n-\mu)'\left(S_n^2+\rho_n I_q\right)^{-1}(f(x,t_i)-\mu)}}{\sum\limits_{t_k\in\mathcal{T}} e^{-(\bar{f}_n-\mu)'(S_n^2+\rho_n I_q)^{-1}(f(x,t_k)-\mu)}},$$

where we recall that $S_n^2 = \frac{1}{n}\sum_{i=1}^{n}\left(f(x_i,t_i)-\mu\right)\left(f(x_i,t_i)-\mu\right)'$.

Just like in paragraph 2.4.1, it is possible to have an estimator of μ based on another corpus and to obtain a plug-in version of this quantity. Thus the problem will essentially be to have an adequate value for the penalization parameter. We will focus on this problem in the next chapter.

2.5 Application

In this section we will build a POS-tagger, which is based on a penalized maximum entropy principle, that takes as input a sentence, and assigns a grammatical class (or POS-tag) to each word in this sentence, using the "penalization" ideas developed above (as well as in the next chapter for the choice of the optimal value of the penalty). To accomplish this, we use the PennTreebank corpus, which uses a tagset containing a total of 46 tags, 36 grammatical tags (verbs, nouns, prepositions, etc.) and 10 punctuation tags (comma, closing brackets, etc.). More precisely, the version of the corpus that we are using, is the one included in the python nltk package. It contains 3914 sentences, which represent 100676 tokens (here single words) or 12408 tokens without repetitions. We extract randomly (several times) a sample of size $N = 10000$ from the 100676 initial tokens for memory capacity reasons.

2.5.1 Preparation of the database

To begin, the first step is to construct two functions:

1) a context function that takes a tagged sentence in the form of (t_i, w_i) pairs (tag, word), $i = 1$ to the size of the sentence (in term of number of words), as input and returns the same sentence but in the form of (t_i, x_i) pairs (tag, context), where x is a context vector that contains information about the word w as well as its neighboring words within the sentence where it was observed. The information we

have retained includes the two words preceding the central word w, the two words following w, whether w is the beginning or end of a sentence, and whether it is a number.

The following table 2.1 provides an example of transforming the following tagged sentence into (tag, context) pairs instead of (tag, word) pairs.

Pierre	Vinken	,	61	years	old	,	will	join	the	board
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
NNP	NNP	,	CD	NNS	JJ	,	MD	VB	DT	NN

as	a	nonexecutive	director	Nov.	29	.
↓	↓	↓	↓	↓	↓	↓
IN	DT	JJ	NN	NNP	CD	.

Tag ($\mathbf{t_i}$)	**Context ($\mathbf{x_i}$)**								
	w_{i-2}	w_{i-1}	$\mathbf{w_i}$	w_{i+1}	w_{i+2}	Digit	Capital	w_0	w_∞
NNP	0	0	Pierre	Vinken	,	0	1	1	0
NNP	0	Pierre	Vinken	,	61	0	1	0	0
,	Pierre	Vinken	,	61	years	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
VB	,	wil	join	the	board	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
NNP	nonexecutive	director	Nov.	29	.	0	0	0	0
CD	director	Nov.	29	.	0	0	1	0	0
.	Nov.	29	.	0	0	0	0	0	1

Table 2.1: An example of (Tag, Context) pairs

- w_i represents the central word
- w_{i-1} and w_{i+1} represent respectively the preceding word and the following word by one position
- w_{i-2} and w_{i+2} represent respectively the preceding word and the following word by two positions
- *Digit* and *Capital* check if the central word is a digit or if it starts with a capital respectively
- w_0 is equal to 1 if the central word is a starting word (the first word of the sentence), 0 otherwise.
- w_∞ is equal to 1 if the central word is an ending word (the last word of the sentence) and 0 otherwise.

For instance the pair $(w_{10}, t_{10}) = (\text{the}, \text{DT})$ is transformed into $(t_{10}, x_{10}) = (\text{DT}, x_{10})$ where $x_{10} = (\text{will}, \text{join}, \text{the}, \text{board}, \text{as}, 0, 0, 0, 0)$.

2) a feature function that takes a context and returns a high-dimensional binary vector. Each component of this vector (or feature) equals 1 if the condition is satisfied and 0 otherwise. To accomplish this, we construct a dictionary of central words, a dictionary of words one position before the central word, a dictionary of words two positions before the central word, and so on. We concatenate these five dictionaries. For a context x_i, we retrieve the "previous word" information. If this word appears in the dictionary of previous words, then the context feature vector will have zero components everywhere except for the position of the word. The conditions are of the form w_i = a particular word from the dictionary of central words, w_{i-1} = a particular word from the dictionary of previous words, etc. That

is we create as many dichotomic variables as there are possible sequences of 5 words and select only the one that occurs more than a given threshold.

For example, if the dictionary of words corresponding to two positions before the central word contains 35 words, and the current context being examined contains information w_{i-2} that appears at the 4th position of this dictionary, then the feature vector block corresponding to the words two positions before the central word for this context will be of the form:

$$(0, 0, 0, 1, 0, 0, \ldots, 0, 0)$$

Actually, the position of the 1 does not only indicate the presence of an information related to the word alone, but to the word combined with a tag, which means that there may be two positions (two features) for the same word but with a different tag. Therefore, it should be understood that the features are functions of the pair $f(x_i, t_i)$ and not just functions of the context $f(x_i)$. In the example of POS-tagging given in the first chapter, we see that the word "flies" can have two possible tags (NN and VB). So, for this same word, there will be two different features in the block of the central word, one that activates only when the central word of the context x_i=flies and t_i =NN, and a second feature that activates when the central word of the context x_i =flies and t_i =VB.

features	...	...	...	(flies, NN)	(flies, VB)	...	...	...	...	...
	↓			↓	↓				↓	↓
f(flies, NN)	0	...	0	1	0	0	0	...	0	0
f(flies, VB)	0	...	0	0	1	0	0	...	0	0

We also construct some features that look at pairs (tag, suffix) where the suffix represents the last three or the last two letters of the central word of a given context. We also perform a filter-based selection of the features to only keep those that are observed more than ten times (since it was sufficient in our case to get good performance, but the filter with a threshold equal to 10 can be modified in other cases).

We now have at our disposal a new dataset in the form of (tag, feature vector) pairs $\{(t_i, f(t_i, x_i))\}_{i\in\{1,\ldots,n\}}$, so everything is ready to start model estimation.

2.5.2 Results

After estimating μ, by the empirical mean

$$\hat{\mu}_N = \frac{1}{N}\sum_{i=1}^{N} f\left(t_i, x_i\right)$$

on the entire initial dataset, we split the dataset into two parts:

- a training sample $\{(t_i, f(t_i, x_i))\}_{i\in\{1,\ldots,n\}}$ (representing 75% of the initial dataset)
- a test sample $\{(t_i, f(t_i, x_i))\}_{i\in\{1,\ldots,n_0\}}$ (25% of the initial dataset).

On the training dataset, we calculate the empirical mean $\bar{f}_n = \frac{1}{n}\sum_{i=1}^{n} f\left(t_i, x_i\right)$ and the empirical covariance matrix of the features

$$S_n^2 = \frac{1}{n}\sum_{i}^{n} \left(f(x_i, t_i) - \hat{\mu}_N\right)\left(f(x_i, t_i) - \hat{\mu}_N\right)'$$

as well as the penalty using the formula proposed in Chapter 3 (in the same spirit of Ledoit and Wolf (2004) [115]), using the modified Frobenius scalar product and its associated norm,

$$\rho_n = \frac{\hat{\beta}_n^2 \hat{\sigma}_n^2}{\hat{\alpha}_n^2} \quad \text{where} \quad \hat{\sigma}_n^2 = \left\langle S_n^2, I_q \right\rangle \ ; \quad \hat{\delta}_n^2 = \left\| S_n^2 - \hat{\sigma}_n^2 I_q \right\|^2 \ ; \quad \hat{\alpha}_n^2 = \hat{\delta}_n^2 - \hat{\beta}_n^2$$

$$\text{with } \bar{\beta}_n^2 = \frac{1}{n^2}\sum_{i=1}^{n}\left\|f(x_i,t_i)(f(x_i,t_i))' - S_n^2\right\|^2 \text{ and } \hat{\beta}_n^2 = \min\left(\bar{\beta}_n^2, \hat{\delta}_n^2\right).$$

This allows us to estimate the conditional probabilities of each tag given the context x as follows

$$\forall t_i \in \mathcal{T}, \quad \hat{p}\left(t_i|x\right) = \frac{e^{-(\bar{f}_n-\hat{\mu}_N)'\left(S_n^2+\rho_n I_q\right)^{-1}(f(x,t_i)-\hat{\mu}_N)}}{\sum\limits_{t_k\in\mathcal{T}} e^{-(\bar{f}_n-\hat{\mu}_N)'(S_n^2+\rho_n I_q)^{-1}(f(x,t_k)-\hat{\mu}_N)}}.$$

Once these probabilities are obtained, the tag assigned to the input context is the one for which the estimated conditional probability is the highest :

$$\forall x_i \in \text{Training set}, \quad \hat{t}_i = \underset{t_k\in\mathcal{T}}{\text{argmax}}\left\{\hat{p}\left(t_k, x_i\right)\right\}.$$

We then estimate the model's error on the test sample by the number of misclassifications :

$$\text{Error} = \frac{1}{n_0}\sum_{i=1}^{n_0}\mathbf{1}_{\left\{t_i\neq\hat{t}_i\right\}} \Leftrightarrow \text{Precision} = \frac{1}{n_0}\sum_{i=1}^{n_0}\mathbf{1}_{\left\{t_i=\hat{t}_i\right\}} = 1 - \text{Error}.$$

The same procedure is repeated 10 times on different random samples of size N=10000 drawn without replacement from the initial dataset containing 100676 entries (tag-context pairs). Finally, we achieve an estimation accuracy of 98% (on average over the different training samples) and a prediction accuracy of 95% (on average over the test samples).

The estimated conditional probabilities

Here is an example of the values of the estimated conditional probabilities for the following sentence which is an observed sentence among the training set:

$$(w_1, \ldots, w_{18}) = \text{Pierre Vinken, 61 years} \ldots \text{Nov. 29.}$$

Let's consider $(x_1, \ldots, x_{18})$ the corresponding contexts to each word $(w_i)_{i=1,\ldots,18}$, i.e.:

Pierre	Vinken	,	61	years	...	nonexecutive	director	Nov.	29	.
↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
$(x_1$	x_2	x_3	x_4	x_5	...	x_{14}	x_{15}	x_{16}	x_{17}	$x_{18})$

The table 2.2 gives the conditional probabilities of a tag given a context. It gives also the predicted POS-tag for each context which is simply the tag with the highest conditional probabilities. The probability values are rounded.

Tag t	**Conditional probabilities** $\mathbb{P}(t\|x_i)$						
	x_1	x_2	x_3	...	x_{16}	x_{17}	x_{18}
NN	0.21	0,02	0,017	...	0.41	0	0
NNS	0	0	0	...	0	0	0
NNP	0.76	0.95	0	...	0.4	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
VBD	0	0	0	...	0	0	0
Predicted POS-tag	NNP	NNP	,	...	NN	CD	.

Table 2.2: Estimated conditional probabilities

We also constructed a similar model that classifies an input text into a simple or complex version. This classifier is presented in the last Chapter.

Annexe

The three following tables that are very useful for our tasks are partly taken from Bertail et al. (2007) [17].

Some Fenchel-Legendre transforms used in generalized MaxEnt or generalized empirical likelihood

$f = g^*$		$g = f^*$	
Fonction	$d(f)$	Fonction	$d(g)$
0	$[-1,1]$	$\mid x\mid$	$\mathbb{R}$
0	$[0,1]$	x^+	$\mathbb{R}$
$\frac{\mid x\mid^p}{p}\ \forall p>1$	$\mathbb{R}$	$\frac{\mid x\mid^q}{q}$ pour $\frac{1}{p}+\frac{1}{q}=1$	$\mathbb{R}$
$-\frac{\mid x\mid^p}{p}\ \forall p\in]0,1[$	$\mathbb{R}_+$	$-\frac{(-x)^q}{q}$	$-\mathbb{R}_+^*$
$\sqrt{(1+x^2)}$	$\mathbb{R}$	$-\sqrt{(1-x^2)}$	$[-1,1]$
$-\log(x)$	$\mathbb{R}_+^*$	$-1-\log(-y)$	$-\mathbb{R}_+^*$
$\log(\cos(x))$	$\left]-\frac{\pi}{2},\frac{\pi}{2}\right[$	$\frac{x}{\tan(x)}-\frac{1}{2}\log(1+x^2)$	$\mathbb{R}$
e^x	$\mathbb{R}$	$\begin{cases} x\log(x)-1 & \text{si } x>0 \\ 0 & \text{si } x=0 \end{cases}$	$\mathbb{R}_+$
$\log(1+e^x)$	$\mathbb{R}$	$\begin{cases} x\log(x)+(1-x)\log(1-x) & \text{si } x\in]0,1[\\ 0 & \text{si } x\in\{0,1\} \end{cases}$	$[0,1]$
$-\log(1-e^x)$	$\mathbb{R}_+^*$	$\begin{cases} x\log(x)-(1+x)\log(1+x) & \text{si } x>0 \\ 0 & \text{si } x=0 \end{cases}$	$\mathbb{R}_+$

Table 2.3: A few Fenchel-Legendre transforms

Cressie-Read

Most divergences used in practice (Kullback, relative entropy, χ^2 and Hellinger among others) are specific cases of the Cressie-Read divergence (see Csiszár 1967). The following table 2.4 gives the functions φ and φ^* for usual examples.

Divergences	α	φ_α		φ^*_α	
		$\varphi_\alpha(x)$	$d(\varphi_\alpha)$	$\varphi^*_\alpha(x)$	$d(\varphi^*_\alpha)$
relative entropy	1	$e^x - 1 - x$	$\mathbb{R}$	$(x+1)\log(x+1) - x$	$]-1, +\infty]$
Kullback	0	$-\log(1-x) - x$	$]-\infty, 1[$	$x - \log(1+x)$	$]-1, +\infty[$
Hellinger	0.5	$\frac{x^2}{2-x}$	$]-\infty, 2[$	$2(\sqrt{(x+1)} - 1)^2$	$]-1, +\infty[$
χ^2	2	$\frac{x^2}{2}$	$\mathbb{R}$	$\frac{x^2}{2}$	$\mathbb{R}$
Cressie-Read	α	$\frac{[(\alpha-1)x+1]^{\frac{\alpha}{\alpha-1}} - \alpha x - 1}{\alpha}$	–	$\frac{(1+x)^\alpha - \alpha x - 1}{\alpha(\alpha-1)}$	–

Table 2.4: Some convex functions corresponding to Cressie-Read divergences

For general Cressie-Read, the domain depends on α as can be seen in the preceding examples.

Optimal weights yielding the corresponding dual likelihood

Divergences	**optimal weights $q_i^\diamond$**	**minimum of the divergence**
relative entropy	$\frac{1}{n}\exp(\lambda^\diamond{}'(X_i-\mu))$	$\sum q_i^\diamond \log(nq_i^\diamond)+1-\sum q_i^\diamond$
Kullback	$\dfrac{1}{n(1-\lambda^\diamond{}'(X_i-\mu))}$	$-1-\sum \frac{1}{n}\log(nq_i^\diamond)+\sum q_i^\diamond$
Hellinger	$\dfrac{4}{n(2-\lambda^\diamond{}'(X_i-\mu))^2}$	$2\sum\left(\sqrt{q_i^\diamond}-\sqrt{\frac{1}{n}}\right)^2$
χ^2	$\frac{1}{n}(1+\lambda^\diamond{}'(X_i-\mu))$	$\sum \dfrac{(nq_i^\diamond-1)^2}{2n}$
Creassie-Read	$\frac{1}{n}\left[1+(\alpha-1)\lambda^\diamond{}'(X_i-\mu)\right]^{\frac{1}{\alpha-1}}$	$\sum_{i=1}^n \frac{(nq_i^\diamond)^\alpha-\alpha nq_i^\diamond+\alpha-1}{n\alpha(\alpha-1)}$

Table 2.5: Weights and minimum of the divergence at the optimum

Chapter 3

Regularized Hotelling's T_n^2 statistics in high dimension

We obtain exponential inequalities for regularized Hotelling's T_n^2 statistics, that take into account the potential high dimensional aspects of the problem. We explore the finite sample properties of the tail of these statistics by deriving exponential bounds for symmetric distributions and also for general distributions under weak moment assumptions (we never assume exponential moments). For this, we use a penalized estimator of the covariance matrix and propose an optimal choice for the penalty coefficient.

3.1 Introduction

In many applications (for instance in genomics or natural language processing), the dimension of the parameter of interest q is large in comparison to the sample size n and sometimes is increasing with n. Consider for instance the problem of estimating or testing a mean of variables in $\mathbb{R}^q$, with $q > n$; in that case, the empirical covariance matrix is not full rank and does not even converge to the true one when n tends to infinity and is ill-conditioned (see Johnstone (2001) [98]). As a consequence, the usual Hotelling's T_n^2 tests in a large dimension framework are no longer valid. It is thus important to construct estimators and testing procedures that take into account the high dimensional aspects of the problem (as done for instance in Ledoit and Wolf (2000, 2022) [115, 116], see also the references therein). One relevant proposition which has been developed in the statistical literature is to use a penalized estimator of the covariance matrix which is non-singular and to use this matrix in tests. In that spirit, Chen et al. (2011) [43] have obtained asymptotically valid regularized Hotelling's T_n^2 tests for the mean in the Gaussian case in a high dimensional framework, when n and $q \equiv q(n)$ tend to infinity at some specific rate. Li et al. (2020) [122] have extended these results to some specific sub-gaussian distribution. The purpose of this chapter is to further explore the finite sample properties of such tests by deriving exponential bounds of some correctly regularized Hotelling's T_n^2 under general distributions, including ones with very few moments.

Such bounds allow to build conservative confidence regions for the parameter of interest. They are also of interest in statistical learning to control risk even with unbounded loss functions. For this, we derive exponential bounds for some regularized Hotelling's T_n^2 statistics in the spirit of Bertail et al (2008) [14], who obtained bounds for self-normalized quadratic forms or the Hotelling's T_n^2 statistic when $q < n$. We show that for symmetric distributions, only moments of order 2 are needed and we only assume the existence of moments of order 8 for general distributions.

Let $Z, Z_1, ..., Z_n$ be i.i.d. centered random vectors with probability distribution P, defined on a probability space $(\Omega, A, \mathbb{P})$ with values in $\left(\mathbb{R}^{q(n)}, \mathcal{B}, P\right)$ endowed with the L_2 norm $\|.\|_2$. We denote $\mathbb{E}$ the expectation under P. Put $Z^{(n)} = (Z_i)_{1 \leq i \leq n}$. As n and $q(n)$ go to infinity, notice that actually

$(Z^{(n)})_n$ defines a triangular array of random variables with varying dimensions. However, since we are interested in finite sample properties, we will drop the dependence in n. In particular, we use q instead of $q(n)$. But keep in mind that q is a function of n in an asymptotic framework. The covariance matrix of the observation is given by $S^2 = \mathbb{E}(ZZ')$, where we denote by Z' the transpose of Z and S the square root of S^2. The sample covariance matrix is defined here by

$$S_n^2\left(Z^{(n)}\right) = \frac{1}{n}\sum_{i=1}^{n} Z_i Z_i'.$$

To simplify notations, we denote the sample covariance matrix of Z_i's by S_n^2 when there is no confusion. Notice that we do not center by the empirical mean.
Denote by

$$\bar{Z}_n = n^{-1}\sum_{i=1}^{n} Z_i.$$

We recall that Hotelling's T_n^2, which can be seen as a quadratic form of self-normalized sums, is given by

$$T_n^2 = n\bar{Z}_n' S_n^{-2}\bar{Z}_n,$$

when $q < n$ and $S_n^{-2} = \left(S_n^2\right)^{-1}$. For some nonnegative real numbers, ρ_1 and ρ_2, define Σ_n^2 the linear combination of the identity matrix with the sample covariance matrix

$$\Sigma_n^2 \equiv \Sigma_n^2(\rho_1, \rho_2) = \rho_1 I_q + \rho_2 S_n^2,$$

with I_q the identity matrix of size q. For $\rho_1 = 0$ and $\rho_2 = 1$, $\Sigma_n^2(0,1) = S_n^2$ is the empirical covariance matrix, which is singular for $q > n$. When $\rho_2 = 1$ and $\rho_1 > 0$ (and small), Σ_n^2 corresponds to a Tikhonov regularization of the sample covariance matrix: see Tikhonov (1963) [192]. It is precisely this estimator which is used in the tests proposed by Chen et al (2011) [43]. However, it is shown in Ledoit and Wolf (2004) [115] that if one chooses adequately ρ_1 and ρ_2 then one can obtain a well-conditioned estimator of the covariance matrix which is invertible and more accurate than the sample covariance for some L_2-distance.
We denote by Σ^2 the expectation of Σ_n^2, which is given by

$$\Sigma^2 \equiv \Sigma^2(\rho_1, \rho_2) = \rho_1 I_q + \rho_2 S^2.$$

Actually, such modification ensures that we can control the distance between Σ_n^2 and S^2: this will be fundamental to obtain exponential bounds.
In the following, we are interested in the Hotelling's T_n^2 statistic with a linear combination of the sample covariance and the identity, that we now call the regularized Hotelling's T_n^2 statistic defined by

$$T_n^2(\rho_1, \rho_2) = n\bar{Z}_n'\Sigma_n^{-2}(\rho_1, \rho_2)\bar{Z}_n$$

generalizing the proposal of Chen et al (2011)[43].

In the framework of high dimension, such quantities also appear naturally when studying empirical likelihood under a lot of constraints, penalized in its dual form by an L_2-norm: see for instance Newey and Smith (2004) [148], Lahiri and Mukhopadhyay (2012), [106], Carrasco and Kontchoni (2017) [35] among others.

When $q < n$, exponential bounds for $T_n^2(0,1)$ (that is, with the empirical covariance matrix instead of a regularized one) have been obtained by Bertail et al (2008) [14]. Their exponential bound is controlled by two terms: (1) an exponential term corresponding to a "Hoeffding" or Pinelis (1994) [157] type of inequality applied to a symmetrized version of the observations and (2) an exponential bound which essentially controls the minimum eigenvalue of the sample covariance matrix and the proximity of S_n^2 to S^2. However, for $q \geq n$ such inequality can not hold since in that case the minimum eigenvalues of S_n^2 is always 0. Moreover, it can easily be seen from the results of [14] that the bound becomes very bad

when $q > n$ or/and when q and n are of the same order. We obtain in this chapter general results with an adequate choice of ρ_1 and ρ_2 when q is bigger than n and when q and n are such that $\frac{q}{n} \to l \in]0, \infty[$.

The chapter is divided into four parts including this introduction. In the second part, we recall some known exponential inequalities for $q < n$ under weak moments assumptions. Then we obtain an oracle exponential inequality for the regularized Hotelling's T_n^2, assuming that the values ρ_1 and ρ_2 are fixed and known. Some interesting sharp bounds which may be useful in statistical learning assuming symmetry are obtained for any n and q large. We then establish a general inequality for $q = O(n)$ for non-symmetric distributions under a few moments' assumptions. In the third part, we estimate the optimal values ρ_1^* and ρ_2^* and show that the inequality remains valid up to some additional small terms controlling the concentration of these estimators around their true value. We illustrate our results with some simulations in the last part.

3.2 Oracle exponential bounds for regularized Hotelling's T_n^2

In the following, we define the penalized Hotelling's T_n^2 as the particular regularized Hotelling's statistic $T_n^2(\rho, 1)$ with $\rho \geq 0$. The aim of this section is to establish an oracle exponential inequality of the distribution of the penalized Hotelling's T_n^2 in the case $q \geq n$ and when the distribution of the data is symmetric (Theorem 3.2.1 and Theorem 3.2.2) as well as in the general case, that is when the distribution is not necessarily symmetric (Theorem 3.2.3).

3.2.1 Known bounds for Hotelling's T_n^2

Some bounds for T_n^2 or self-normalized sums may be quite easily obtained in the symmetric case (that is for random variables having a symmetric distribution see Pinelis (1994) [157]) and are well-known in the unidimensional case $q = 1$. In non-symmetric and/or multidimensional cases with $q < n$, these bounds are new and not trivial to prove. One of the main tools for obtaining exponential inequalities in various settings is the famous Hoeffding inequality (see Hoeffding (1994) [93]). For centered independent real random variables $Y_1, \ldots, Y_n$, that are bounded, say $|Y_i| < 1$, for all $i \in \{1, \ldots, n\}$, we have, for $a_i \in [-1, 1]$ such that $\sum a_i^2 = 1$,

$$\forall t > 0, \quad \mathbb{P}\left(\left(\sum_{i=1}^n a_i Y_i\right)^2 \geq t\right) \leq 2\exp\left(-\frac{t}{2}\right).$$

A direct application of this inequality to self-normalized sums (via a randomization step introducing independent Rademacher r.v.'s ε_i) yields that, for independent real ($q = 1$) symmetric random variables Z_i, $i \in \{1, \ldots, n\}$ and not necessarily bounded (nor identically distributed). Indeed, we have by putting $Y_i = \varepsilon_i$ and $a_i = \frac{Z_i}{\left(\sum Z_i^2\right)^{1/2}}$

$$\begin{aligned}
\forall t > 0, \quad \mathbb{P}\left(T_n^2 \geq t\right) &= \mathbb{P}\left(\frac{\left(\sum_{i=1}^n Z_i\right)^2}{\sum_{i=1}^n Z_i^2} \geq t\right) = \mathbb{P}\left(\frac{\left(\sum_{i=1}^n Z_i\varepsilon_i\right)^2}{\sum_{i=1}^n Z_i^2} \geq t\right) \\
&= \mathbb{E}\left[\mathbb{P}\left(\frac{\left(\sum_{i=1}^n Z_i\varepsilon_i\right)^2}{\sum_{i=1}^n Z_i^2} \geq t \;\middle|\; (Z_i)_{i=1,..n}\right)\right] \\
&\leq 2\exp\left(-\frac{t}{2}\right).
\end{aligned}$$

Pinelis (1994) [157] has obtained with a different technique, a sharp χ^2 type of bounds which generalizes this kind of results for multivariate data when $q < n$. He proved that, if Z has a symmetric distribution, without any moment assumption on the variables Z_i, then one has

$$\forall t > 0, \quad \mathbb{P}\left(T_n^2 \geq t\right) \leq \frac{2e^3}{9}\overline{F}_q(t), \tag{3.1}$$

where $F_q(t)$ is the cumulative distribution function (cdf) of a $\chi^2(q)$ distribution. The density is denoted by f_q. A crude approximation yields that for t large enough,

$$\mathbb{P}\left(T_n^2 \geq t\right) \leq \frac{e^3}{9} \frac{2^{2-\frac{q}{2}}}{\Gamma(\frac{q}{2})} t^{\frac{q}{2}-1} \exp(-t/2),$$

where $\Gamma(x) = \int_0^{+\infty} t^{x-1} e^{-t} dt$ is the gamma function.

Notice that, for $q = 1$ this bound (only valid for large t) is better than the crude Hoeffding bound since we recover the missing factor $\frac{1}{\sqrt{t}}$ in front of the exponential (see Talagrand (1995) [187]). When $q < n$, using a multidimensional version of Panchenko's symmetrization lemma (Panchenko (2003) [152]) Bertail et al. (2008) [14] have obtained an exponential inequality for the general distribution of Z with finite kurtosis $\gamma_4 = \mathbb{E}\left(\left\|S^{-1}Z\right\|_2^4\right)$. More precisely, they establish that under $0 < \gamma_4 < \infty$,
(i) for $t > nq$, $\mathbb{P}\left(T_n^2 \geq t\right) = 0$.
(ii) for any $a > 1$, and any nonnegative t such that $2q(1+a) \leq t \leq nq$, the following bound holds:

$$\begin{aligned} \mathbb{P}(T_n^2 \geq t) \quad \leq \quad & \frac{2e^3}{9\Gamma(\frac{q}{2}+1)} \left(\frac{t-q(1+a)}{2(1+a)}\right)^{\frac{q}{2}} \exp\left(-\frac{t-q(1+a)}{2(1+a)}\right) \\ & + C(q) n^{3-\frac{6}{q+1}} \exp\left(-\frac{n\left(1-\frac{1}{a}\right)^2}{\gamma_4(q+1)}\right), \end{aligned}$$

where $C(q)$ is an explicit constant.

The first term is essentially equivalent to the tail of a $\chi^2(q)$ distribution (up to an explicit constant), while the second term controls the speed of convergence of S_n^2 to S^2, when $\gamma_4 < \infty$. The constant a controls the balance between these two terms on the right-hand side of the inequality and may be optimized. Notice that this second exponential term is small when $q << n$ but explodes in n^3 if $q/n \to l > 0$ for large n, making this bound totally useless in that case.

In the general multidimensional framework considered in Bertail et al (2008) [14] and in this paper, the main difficulty is to keep the self-normalized structure when symmetrizing the initial sum. In the next sections, the results of Bertail et al. (2008) [14] obtained for $q < n$ are extended to the case $q \geq n$ by using a regularized version of S_n^2. This inequality is based on an appropriate diagonalization of the regularized sample covariance matrix which allows applying Pinelis (1994)'s inequality [157] (see section 3.2.2). This crude inequality is refined in section 3.2.3. When dealing with the general case (see section 3.2.4), we establish first a multivariate symmetrization lemma 3.4.2 in the spirit of Panchenko (2003) [152]. This symmetrization partially destroys the self-normalized structure (the normalization is then $\Sigma_n^2 + \Sigma^2 = 2\Sigma_n^2 + (\Sigma^2 - \Sigma_n^2)$ instead of the expected normalization Σ_n^2), but the right standardization can be recovered (up to the factor 2) by obtaining a lower tail control of the distance between Σ_n^2 and Σ^2. To control this distance and make it as small as possible we will use the results of Ledoit and Wolf (2004) [115].

3.2.2 Bounds for regularized Hotelling's T_n^2 in a symmetric framework

We now obtain a simple inequality for the regularized Hotelling's T_n^2 in the symmetric case, based on previous results by Pinelis (1994) [157]. It essentially shows that the tail of the regularized Hotelling's T_n^2 is controlled by the tail of a $\chi^2(n)$ distribution.

Theorem 3.2.1. *Assume that Z has a symmetric distribution with finite covariance matrix then,*

without any additional moment assumption, we have, for any $n > 1$, for $t > n$, for any $\rho_1, \rho_2 > 0$,

$$\begin{aligned}\mathbb{P}\left(T_n^2\left(\frac{\rho_1}{\rho_2}, 1\right) \geq t\right) &= \mathbb{P}\left(n\bar{Z}_n'\Sigma_n^{-2}\left(\rho_1, \rho_2\right)\bar{Z}_n \geq \frac{t}{\rho_2}\right) \\ &\leq \frac{2e^3}{9}\bar{F}_n\left(t\right) \\ &\leq \frac{2e^3}{9}\exp\left(-\frac{(t-n)^2}{4t}\right), \end{aligned} \tag{3.2}$$

where F_n is the cdf of a $\chi^2(n)$ distribution.
Moreover, for any $\rho > 0$, we have

$$\begin{aligned}\mathbb{P}\left(\frac{T_n^2\left(\rho, 1\right) - n}{\sqrt{2n}} \geq t\right) &= \mathbb{P}\left(\frac{n\bar{Z}_n'\Sigma_n^{-2}\left(\rho, 1\right)\bar{Z}_n - n}{\sqrt{2n}} \geq t\right) \\ &\leq \frac{2e^3}{9}\exp\left(\frac{-t^2}{2\left(1 + \sqrt{2}\frac{t}{\sqrt{n}}\right)}\right). \end{aligned} \tag{3.3}$$

The inequality (3.2) yields a control of $T_n^2(\rho_1, \rho_2) = n\bar{Z}_n'\Sigma_n^{-2}\left(\rho_1, \rho_2\right)\bar{Z}_n$, when using a linear shrinkage estimator of the variance. This in turn can be simplified in (3.3), to a truly penalized Hotelling's T_n^2. Note that for any $\rho_1, \rho_2 > 0$,

$$\frac{\Sigma_n^2\left(\rho_1, \rho_2\right)}{\rho_2} = \frac{\rho_2 S_n^2 + \rho_1 I_q}{\rho_2} = S_n^2 + \frac{\rho_1}{\rho_2}I_q$$

and for any $\rho > 0$,

$$\Sigma_n^2\left(\rho, 1\right) = S_n^2 + \rho I_q$$

is a penalized estimator of the covariance matrix. Inequality (3.3) can be interpreted as a Bernstein-type inequality.

Remark: These inequalities hold for any choice of ρ_1 and ρ_2. However for the inequalities to be sharp, ρ_1 and ρ_2 should be chosen adequately. First from the proof of Theorem 3.2.1, we see that the inequality is sharp only when ρ_1 is close to 0, which is in accordance with what we know about Tikhonov regularisation (1963) [192]. Actually when ρ_1 tends to $0, \Sigma_n^{-2}\left(\rho_1, \rho_2\right)$ is going to be identical to $\frac{1}{\rho_2}(S_n^2)^-$ where $(A)^-$ is the Moore-Penrose or generalized inverse of A (which is unique for symmetric matrices). Notice that the proof of the theorem and the inequality remain valid if we use $n\bar{Z}_n'\left(S_n^2\right)^-\bar{Z}_n$ rather than $n\bar{Z}_n'\Sigma_n^{-2}\left(\rho_1, \rho_2\right)\bar{Z}_n$. In the procedure of Chen et al. (2011) [43] this means that asymptotically there is no difference between standardizing by the regularized variance or by the generalized inverse of the covariance matrix. The regularization just serves as a trick to approximate the generalized inverse. However, the finite sample properties of the regularized Hotelling's T_n^2 will strongly depend on the choice of ρ_1 and ρ_2.

3.2.3 An improved bound for penalized Hotelling's T_n^2 in the symmetric case

It can be seen from the proof of Theorem 3.2.1 that the penalized Hotelling's T_n^2 statistic essentially behaves like a weighted sum of asymptotically χ^2 random variables. This also explains the results of Chen et al. (2015) [43]. Actually, we can obtain a bound for this quantity relying on the results of Pinelis (1994) [157] and Laurent and Massart (2000) [110] (see p.24 of their paper) who control the tail of the weighted sum of independent $\chi^2(1)$ random variables.
Let $\lambda = (\lambda_j)_{j=1,\ldots,q} \in \mathbb{R}_+^q$ be the eigenvalues of S_n^2 (ordered in a increasing order). We define for any

$\rho_1, \rho_2 > 0$, the following effective dimensions (see [43] for other expressions of these quantities):

$$\Theta_1(\lambda, \rho_1, \rho_2) = \sum_{j=1}^{\inf(n,q)} \frac{\lambda_j}{\rho_1 + \rho_2 \lambda_j}$$

$$\Theta_2(\lambda, \rho_1, \rho_2) = \sqrt{\sum_{j=1}^{\inf(n,q)} \frac{\lambda_j^2}{(\rho_1 + \rho_2 \lambda_j)^2}}$$

$$\Theta_\infty(\lambda, \rho_1, \rho_2) = \sup_{1 \le j \le \inf(n,q)} \left(\frac{\lambda_j}{\rho_1 + \rho_2 \lambda_j} \right).$$

In the next result, we obtain a sharp bound for regularized and penalized Hotelling's T_n^2. Notice that, in that case, the recentering factor depends on $\Theta_1(\lambda, \rho_1, \rho_2)$ and is random. In the proof of Theorem 3.2.1, this value is essentially bounded by n/ρ_2, which is a very bad approximation when ρ_2 is small. Theorem 3.2.2 tells that, for $q \ge n$, the tail of the regularized Hotelling's T_n^2 statistic behaves as the weighted sum of n independent $\chi^2(1)$ r.v.'s where the weights are given by the random factors $\frac{\lambda_j}{\rho_1 + \rho_2 \lambda_j}$. We get some Bernstein bounds for this weighted sum by first randomizing by some independent Gaussian r.v.'s, then conditioning on the data and applying Laurent and Massart (2000)'s Bernstein inequality [110]. This inequality in turn can be transformed into some exact bounds for the statistics of interest.

Theorem 3.2.2. *Assume that Z has a symmetric distribution then, without any moment assumption, we have, for any $n > 1$ and $q > 0$, for any $t > 0$ and for any $\rho_1, \rho_2 > 0$,*

$$\mathbb{P}\left(\frac{T_n^2(\rho_1, \rho_2) - \Theta_1(\lambda, \rho_1, \rho_2)}{\sqrt{2\Theta_2(\lambda, \rho_1, \rho_2)^2}} \ge \sqrt{2}\left(\sqrt{t} + \frac{\Theta_\infty(\lambda, \rho_1, \rho_2)}{\Theta_2(\lambda, \rho_1, \rho_2)} t \right) \right) \le C \exp(-t).$$

with $C = 3824$.
Or equivalently, we have for the penalized Hotelling's statistic, for $n > 1$ and $q > 0$, for any $t > 0$ and, for any $\rho > 0$,

$$\mathbb{P}\left(\frac{T_n^2(\rho, 1) - \Theta_1(\lambda, \rho, 1)}{\Theta_2(\lambda, \rho, 1)} \ge \sqrt{2t} + \frac{\Theta_\infty(\lambda, \rho, 1)}{\Theta_2(\lambda, \rho, 1)} t \right) \le C \exp\left(-\frac{t}{2} \right).$$

In the symmetric case, this theorem enables us to easily obtain confidence regions of level $1 - \delta$, for $\delta \in [0, 1]$ for the regularized Hotelling's statistic, as stated in the following corollary.

Corollary 3.2.1. *Put $c(\delta) = \log \frac{C}{\delta}$ with $C = 3824$. Then, for any $n > 1$ and $q \ge 1$, for any $t > 0$ and for any $\rho_1, \rho_2 > 0$, with probability $1 - \delta$, we have*

$$T_n^2(\rho_1, \rho_2) \le \Theta_1(\lambda, \rho_1, \rho_2) + 2\Theta_2(\lambda, \rho_1, \rho_2) \left(\sqrt{c(\delta)} + \frac{\Theta_\infty(\lambda, \rho_1, \rho_2)}{\Theta_2(\lambda, \rho_1, \rho_2)} c(\delta) \right),$$

The proof of this corollary is left to the reader. This result holds for any n and q. When $q \le n$ is large, we can actually put $\rho_1 = 0$ and get some Pinelis' type bounds (when the χ^2 distribution tail is itself approximated by a Gaussian tail).

The constant C comes from a result of Chasapis and al (2022) [41] who extended a result of Pinelis [157] (1994). Indeed they state that, when symmetrizing, for smooth functions of quadratic forms, Rademacher variables may be replaced by standard normal variables. However, their constant is clearly not optimal and we expect the optimal C to be $2e^3/9$ as in Pinelis [157] (1994).

The bounds in Theorem 3.2.2 and Corollary 3.2.1 can be used in practice for testing purposes in particular in anomaly detection in statistical learning. See for instance the literature on intrusion detection systems using multivariate control charts based on Hotelling's T_n^2 (for instance Tracy et al. (1992) [195] and further works by these authors).

3.2.4 Bounds for regularized Hotelling's T_n^2 for non symmetric distribution

We now consider Z with a general (not necessarily symmetric) distribution. We will later prove a symmetrization lemma that generalizes the one obtained in Bertail et al. (2008) [14]. In the following, we also use the results of Ledoit and Wolf (2004) [115] to optimally control the distance between $\Sigma_n^2(\rho_1, \rho_2)$ and S^2. For this, consider the modified Frobenius scalar product between matrices and the corresponding norm given by

$$\langle A, B \rangle = \frac{Tr(AB')}{q}$$

and

$$\|A\|^2 = \langle A, A \rangle = \frac{Tr(AA')}{q}. \tag{3.4}$$

Note that dividing the standard Frobenius scalar product by q enables the norm of the identity I_q to be equal to 1, which is more convenient. In the following, we extend this modified Frobenius norm to vectors by considering, for any vector $Z \in \mathbb{R}^q$,

$$\|Z\|^2 = Tr(ZZ')/q.$$

Additional notations and hypotheses

Put $S^2 = (\sigma_{kj})_{1 \leq k,j \leq q}$ and consider Λ^2 the diagonal matrix of the eigenvalues of S^2 and O the matrix of associated eigenvectors. The eigenvalues are denoted $\mu_1, \ldots, \mu_q$ with $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_q$. We have $S^2 = O'\Lambda^2 O$. Now, for $i \in \{1, \ldots, n\}$, we define

$$Y_i = OZ_i$$

with $Y_i = (Y_{i,1}, \ldots, Y_{i,q})'$.
In order to provide a well-conditioned estimator for large dimensional covariance matrices, Ledoit and Wolf (2004) [115] have studied the minimum of $\mathbb{E}\left(\left\|\Sigma_n^2(\rho_1, \rho_2) - S^2\right\|^2\right)$. This minimization can be seen as a projection problem in the Hilbert space of random matrices, equipped with the inner product $\langle A, B \rangle_{\mathcal{H}} = \mathbb{E}[\langle A, B \rangle]$ with associated norm $\|.\|_{\mathcal{H}}^2 = \mathbb{E}\|.\|^2$.
We assume the four following assumptions:

(A_1) $\exists K_0, K_1 > 0$ such that, for any n and any $q \geq n$, $K_0 \leq \frac{q}{n} \leq K_1$.

(A_2) $\exists K_2 > 0$ such that, for any n and any $q \geq n$, $\frac{1}{q}\sum_{j=1}^q \mathbb{E}\left[Y_{1,j}^8\right] \leq K_2$.

(A_3) $\exists K_3 > 0$ such that for any n and any $q \geq n$, $\frac{1}{K_3} < \mu_1 \leq \mu_q < K_3$.

(A_4) $\exists K_4 > 0$ such that for any n and any $q \geq n$,

$$\nu = \frac{q^2}{n^2} \times \frac{\sum_{(i,j,k,l) \in \mathbf{Q}} \left(\mathrm{Cov}\left(Y_{1,i}Y_{1,j}, Y_{1,k}Y_{1,l}\right)\right)^2}{Card(\mathbf{Q})} \leq \frac{K_4}{n},$$

where $\mathbf{Q}$ denotes the set of all the quadruples that are made of four distinct integers between 1 and q.

Remarks: (A_2) and (A_4) are already assumed in Ledoit and Wolf (2004) [115]. First assumption (A_1) essentially means that $q = q(n)$ is of the same order as n. (A_2) states that the moment of order 8 is bounded in average: this condition holds if the following moment of order 8, $\frac{1}{q}\sum_{j=1}^q \mathbb{E}\left[Z_{1,j}^8\right]$ is finite (by sub-multiplicative inequality and the fact that $\|O\| = 1$). This is a weak condition: we do not require exponential moments and allow for fat tail behavior of the sample. (A_3) ensures that the largest and the smallest eigenvalue of the true covariance matrix are bounded. This rules out the case

when the components of the vector Z are too correlated: consider for instance the degenerate case where S^2 is a matrix full of 1, then in that case the smallest eigenvalue is 0 and the largest is q. The case of a vector with long memory components is studied in Merlevède et al. (2019) [140] : they show that the largest eigenvalue is unbounded. Thus this case does not enter our framework. Assumption (A_4) is immediate in the Gaussian case, since $\nu = 0$ because of the rotation which makes the $Y_{1,j}$'s $j \in \{1, \dots, q\}$ independent. Obviously, for $(Z_{1,j})_j$ independent, $\nu = 0$ as well. More generally if the components of the vector satisfy some adequate α-mixing conditions, then the sum in the hypothesis (A_4) can be seen as a sum of cumulants and may also be controlled using the arguments of Doukhan and León (1989) [64].

Inequalities for random variables with a general distribution

The next Theorem 3.2.3 extends Theorem 3.2.1 to general distributions which are not necessarily symmetric. From now on, following Ledoit and Wolf (2004) [115], we denote ρ_1^* and ρ_2^* the optimal values defined as the minimum arguments of $\mathbb{E}\left\|\Sigma_n^2(\rho_1, \rho_2) - S^2\right\|^2$. Ledoit and Wolf (2004) [115] have obtained

$$\rho_1^* = \frac{\beta^2}{\delta^2}\sigma^2 \quad \text{and} \quad \rho_2^* = \frac{\alpha^2}{\delta^2},$$

with

$$\sigma^2 = \left\langle S^2, I_q \right\rangle ; \quad \alpha^2 = \left\|S^2 - \sigma^2 I_q\right\|^2 ; \quad \beta^2 = \mathbb{E}\left\|S_n^2 - S^2\right\|^2$$
$$\text{and} \quad \delta^2 = \alpha^2 + \beta^2 = \mathbb{E}\left\|S_n^2 - \sigma^2 I_q\right\|^2.$$

Now, we define, for $\alpha^2 \neq 0$,

$$\rho^* = \frac{\rho_1^*}{\rho_2^*} = \frac{\beta^2}{\alpha^2}\sigma^2,$$

which yields the optimal penalized estimator of S_n^2:

$$\Sigma_n^{*2} = \frac{\Sigma_n^2(\rho_1^*, \rho_2^*)}{\rho_2^*} = S_n^2 + \rho^* I_q.$$

If $\alpha^2 = 0$, take $\Sigma_n^{*2} = \sigma^2 I_q$ (in that case we will just need to estimate σ^2).

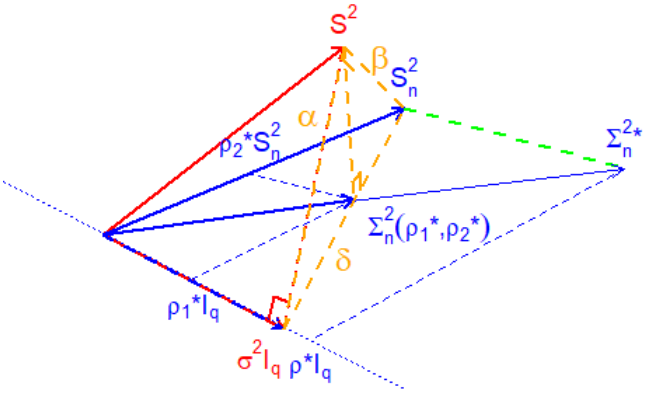


Figure 3.1: **True covariance S^2, sample covariance S_n^2, and $\Sigma_n^2(\rho_1^*, \rho_2^*), \Sigma_n^{2*}$ respectively regularized and penalized sample covariance**

.

In Figure 3.1, the scalar product is $\langle , \rangle_{\mathcal{H}}$ with its associated norm. We represent $\Sigma_n^2(\rho_1^*, \rho_2^*)$, the optimal combination of S_n^2 and I_q defined by orthogonal projection of the true covariance matrix S^2 on

the random vector-space generated by S_n^2 and I_q. Thus $\Sigma_n^{2*} = \Sigma_n^2(\rho^*, 1)$ is the penalization of S_n^2 by I_q with $\rho^* = \frac{\rho_1^*}{\rho_2^*}$. The green dashed line represents the set of penalized estimators $\Sigma_n^2(\rho, 1)$ for which we obtain universal bounds in Theorem 3.2.3.

Theorem 3.2.3. *Assume that Z has a general distribution with finite variance S^2. Assume in addition that assumptions (A_1) to (A_3) hold. Then we have, for any $n > 1$, for any $q \geq n$, and for $t > 2n$,*

$$\mathbb{P}\left(T_n^2(\rho^*, 1) \geq (1 + a^*)\, t\right) = \mathbb{P}\left[n\bar{Z}_n' \Sigma_n^{*-2} \bar{Z}_n \geq (1 + a^*)\, t\right] \quad \leq \quad \frac{2e^3}{9}\left(\frac{t-n}{2}\right)^{\frac{n}{2}} \frac{\exp\left(-\frac{t-n}{2}\right)}{\Gamma\left(\frac{n}{2}+1\right)},$$

with $a^* = 1 + \frac{K_3}{\rho^*}$.

Remark: Here the bounding function for large t behaves like a centered $\chi^2(n)$ distribution, up to the factor $\frac{2e^3}{9}$. The term $(1 + a^*)$ ensures that the smallest eigenvalue of Σ_n^{*2} does not contribute to the inequality.
Notice that the inequality is still valid when using Σ_n^2, the regularized version of S_n^2 instead of the penalized version Σ_n^{*2}, up to a small modification of the bound $(1 + a^*)t$ by the factor $1/\rho_2^*$: for $n > 1$, $q \geq n$, for any $t > 2n$

$$\mathbb{P}\left(T_n^2\left(\rho_1^*, \rho_2^*\right) \geq \frac{1}{\rho_2^*}(1 + a^*)\, t\right) \leq \frac{2e^3}{9}\left(\frac{t-n}{2}\right)^{\frac{n}{2}} \frac{\exp\left(-\frac{t-n}{2}\right)}{\Gamma\left(\frac{n}{2}+1\right)}.$$

3.3 Inequality with estimated parameters

We have proved an exponential inequality for the penalized Hotelling's T_n^2 with theoretical values a^* and ρ^*. In practice these values are unknown. In this section, we estimate these quantities and obtain an inequality for the penalized Hotelling's T_n^2 with estimated parameters. We first recall several results of Ledoit and Wolf (2004) [115] on the asymptotic behavior of regularized empirical covariance estimator Σ_n^2. Lemma 3.3.1 and proposition 3.3.1 below summarize these results with our notations and are proved by Ledoit and Wolf (2004) [115] in different lemmas and a theorem of their paper.
We use the same assumptions as in Ledoit and Wolf (2004) [115]: $\xrightarrow{L_4}$ denotes the fourth-moment convergence as n goes to infinity, i.e.

$$U_n \xrightarrow{L_4} U \Longleftrightarrow \mathbb{E}\left[(U_n - U)^4\right] \underset{n\to\infty}{\longrightarrow} 0.$$

Ledoit and Wolf (2004) [115] essentially have shown that L_4-consistent estimators for σ^2, α^2, β^2 and δ^2 are simply given by their empirical counterparts that is

$$\begin{aligned} \hat{\sigma}_n^2 &= \left\langle S_n^2, I_q \right\rangle \\ \hat{\delta}_n^2 &= \left\| S_n^2 - \hat{\sigma}_n^2 I_q \right\|^2 \\ \hat{\alpha}_n^2 &= \hat{\delta}_n^2 - \hat{\beta}_n^2 \end{aligned}$$

with $\bar{\beta}_n^2 = \frac{1}{n^2}\sum_{i=1}^n \left\| Z_i(Z_i)' - S_n^2 \right\|^2$ and $\hat{\beta}_n^2 = \min(\bar{\beta}_n^2, \hat{\delta}_n^2)$.

Lemma 3.3.1. *[Ledoit and Wolf (2004) [115] lemma 3.2, lemma 3.3, lemma 3.4, lemma 3.5] Under assumptions (A_1) to (A_4), we have*

1. *σ^2, α^2 and β^2 remain bounded (as n and q tend to ∞).*
2. *For all n, $\mathbb{E}\left[\hat{\sigma}_n^2\right] = \sigma^2$, and $\hat{\sigma}_n^2 - \sigma^2 \xrightarrow{L_4} 0$ and $\hat{\sigma}_n^4 - \sigma^4 \xrightarrow{L_4} 0$.*

3. $\hat{\delta}_n^2 - \delta^2 \xrightarrow{L_4} 0$.

4. $\bar{\beta}_n^2 - \beta^2 \xrightarrow{L_4} 0$ *and* $\hat{\beta}_n^2 - \beta^2 \xrightarrow{L_4} 0$.

5. $\hat{\alpha}_n^2 - \alpha^2 \xrightarrow{L_4} 0$.

After replacing the unobservable scalars σ^2, α^2, β^2 and δ^2 by their sample counterparts in the formula of Σ_n^2, Ledoit and Wolf (2004) [115] obtained an estimation of the regularized empirical covariance matrix say

$$\hat{\Sigma}_n^2 = \frac{\hat{\beta}_n^2}{\hat{\delta}_n^2}\hat{\sigma}_n^2 I_q + \frac{\hat{\alpha}_n^2}{\hat{\delta}_n^2} S_n^2.$$

Ledoit and Wolf (2004) [115] have shown that $\hat{\Sigma}_n^2$ and Σ_n^2 are asymptotically equivalent in the modified Frobenius norm.

Proposition 3.3.1. *[Ledoit and Wolf (2004) [115], Theorem 3.2] Under the assumptions* (A_1)*-*(A_4)*, we have*

1. $\lim\limits_{n\to\infty} \mathbb{E}\left\|\hat{\Sigma}_n^2 - \Sigma_n^2\right\|^2 = 0.$

2. *Moreover,* $\hat{\Sigma}_n^2$ *has the same asymptotic expected loss (or risk) as* Σ_n^2 *i.e.*

$$\lim_{n\to\infty} \mathbb{E}\left\|\hat{\Sigma}_n^2 - \Sigma^2\right\|^2 - \mathbb{E}\left\|\Sigma_n^2 - \Sigma^2\right\|^2 = 0.$$

In the same way as Ledoit and Wolf (2004) [115] we define the optimal coefficients ρ_1^* and ρ_2^*. They are estimated respectively by $\hat{\rho}_1^*$ and $\hat{\rho}_2^*$, where $\hat{\rho}_1^* = \frac{\hat{\beta}_n^2}{\hat{\delta}_n^2}\hat{\sigma}_n^2$ and $\hat{\rho}_2^* = \frac{\hat{\alpha}_n^2}{\hat{\delta}_n^2}$. Now, if $\hat{\alpha}_n^2 \neq 0$, we introduce $\hat{\Sigma}_n^{2*}$ the "estimated optimal" penalized version of S_n^2 given by

$$\hat{\Sigma}_n^{*2} = \Sigma_n^2\left(\frac{\hat{\rho}_1^*}{\hat{\rho}_2^*}, 1\right) = S_n^2 + \hat{\rho}_n^* I_q, \quad \text{where } \hat{\rho}_n^* = \frac{\hat{\beta}_n^2\hat{\sigma}_n^2}{\hat{\alpha}_n^2}.$$

Similarly the unobservable threshold constant a^* introduced in theorem 3.2.3 is estimated by $\hat{a}_n^* = 1 + \frac{K_3}{\hat{\rho}_n^*}$. The principle in Figure 3.2 is similar to the one in Figure 1 except that $\hat{\Sigma}_n^2$ is determined first so that the regularized estimator belongs to the yellow line and the optimal estimator $\Sigma_n^2 = \Sigma_n^2(\hat{\rho}_1^*, \hat{\rho}_2^*)$ is the closest value to S^2 on this line. This difference induces an additional error term in our inequalities. Theorem 3.3.1 establishes an exponential bound for the penalized self-normalized sums, when Σ_n^{*2} is replaced by the estimator $\hat{\Sigma}_n^{2*}$ and a^* by $\hat{a}_n^*$, up to a small error term that we control explicitly.

Theorem 3.3.1. *Under the assumptions* (A_1) *to* (A_4)*, we have, for any* $n > 1$*, for any* $q > n$*, for any* $t > 2n$ *and for any small value of* $\epsilon > 0$*,*

$$\begin{aligned}\mathbb{P}\left(T_n^2(\widehat{\rho}_n^*, 1) \geq t\left(1 + \hat{a}_n^* + 2\epsilon\right)\right) &= \mathbb{P}\left(n\bar{Z}_n'\hat{\Sigma}_n^{*-2}\bar{Z}_n \geq t\left(1 + \hat{a}_n^* + 2\epsilon\right)\right) \\ &\leq \frac{2e^3}{9}\left(\frac{t-n}{2}\right)^{\frac{n}{2}} \frac{e^{-\frac{t-n}{2}}}{\Gamma\left(\frac{n}{2}+1\right)} + \frac{C(\epsilon)}{n\epsilon}, \end{aligned} \tag{3.5}$$

where $\hat{a}_n^* = 1 + \frac{K_3}{\hat{\rho}_n^*}$*, and* $C(.)$ *is a real nonnegative function, independent of* n*, defined by*

$$\begin{aligned} C(\epsilon) &= 4K_1\sqrt{K_2}\left(2 + \frac{1}{q} + K_1\right) + 2K_1 G\left(\sqrt{\frac{\epsilon}{2K_1}}\right) \\ &\quad + \frac{4K_1^2\sigma^4}{\epsilon} G\left(\frac{\epsilon}{2\sigma^2 K_1}\right) + \frac{K_3^2}{\epsilon} G\left(\frac{\epsilon}{K_3}\right). \end{aligned}$$

The function G *is defined explicitly in lemma 3.4.6. Notice that* $C(\epsilon)/\epsilon$ *explodes when* ϵ *goes to 0.*

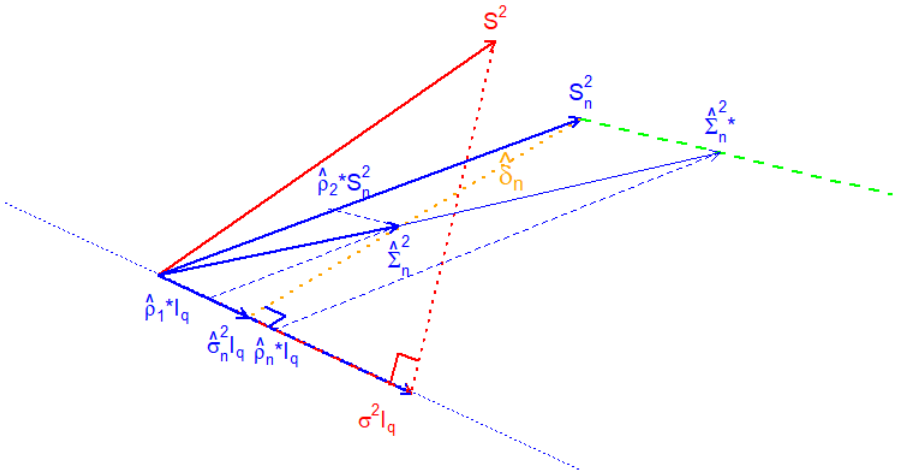


Figure 3.2: **True covariance S^2, sample covariance S_n^2, regularized and penalized estimators of S_n^2, respectively $\hat{\Sigma}_n^2$ and $\hat{\Sigma}_n^{*2}$.**

These results essentially show that we have a $\chi^2(n)$ control in the tail of the distribution, for a threshold larger than $2n(1+\hat{a}_n^*)$ (recall that $2n$ is the variance of a $\chi^2(n)$ distribution). The loss $(1+\hat{a}_n^*)$ is essentially due to the correlation between the components of Z and the deviation from homoscedasticity. The value of ϵ can not be too small but can be optimized by balancing the two terms in the inequality. For a given ϵ and a given level δ it is possible to solve numerically the second term of the inequality (3.5) equal to delta to get a valid bound for the Hotelling's T^2 for any n and q.

3.4 Simulations

In this section, we explore graphically for different distributions, how the dependence structure of the observations and the distance to the homoscedastic framework impact the penalization constants and the tail of the distribution
We generate Gaussian random variables with a given covariance structure corresponding respectively to the following scenarios:

- **scenario a)** the components $Z_{i,j}, j \in \{1,\dots,q\}$ are independent with variance $\sigma_{j,j}$, that is Z_i are i.i.d $N(0,S^2)$ with $S^2 = diag(\sigma_{j,j})_{1\le j\le q}$ for $i \in \{1,\dots,n\}$. The $\sigma_{j,j}$ are themselves generated randomly in a $LN(0,\eta^2)$. We actually expect the variance of the eigenvalues to have a strong influence on the penalized term. The variance η^2 is calibrated for comparison with the dependent case and chosen equal to $log(1+\sqrt{1+4\alpha^2}) - log(2)$ to ensure that the distance between S^2 and $\sigma^2 I_q$ is indeed equal to α^2 (which is chosen the same in the dependent case).

- **scenario b)** the r.v. Z_i's are i.i.d $N(0,S^2)$ with S^2 given by a Toeplitz matrix of the form

$$S^2 = \begin{pmatrix} 1 & s & s^2 & \dots & s^{q-2} & s^{q-1} \\ s & 1 & s & \ddots & \ddots & s^{q-2} \\ s^2 & s & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 & \ddots & s^2 \\ s^{q-2} & \ddots & \ddots & \ddots & 1 & s \\ s^{q-1} & s^{q-2} & \dots & s^2 & s & 1 \end{pmatrix}.$$

Up to a constant, this is the covariance matrix of a stationary $AR(1)$ process with auto-regressive parameter s. This parameter s is thus a dependence parameter in $]-1,1[$ allowing the components of the observations to exhibit more or less dependence.

Notice that in our framework the quantity α^2 is a measure of the complexity of the problem. Actually, if $\alpha^2 = 0$, we can directly use the identity matrix instead of the empirical variance and there is no need for penalizing. For this reason, we are going to compare our simulation results for some given fixed values of α^2 respectively in the dependent and independent cases. For that, we now consider four simulation cases:

(i) scenario a) with α^2 close to 1.10 (note that actually the value of α^2 depends on q but is close to this value in all simulations) corresponding to a standard deviation $\eta = 0.71$;

(ii) scenario b) with the same values of α^2 as in (i) corresponding to a dependence parameter $s = 0.6$;

(iii) scenario a) with α^2 equal respectively to 35.74, 55.63, 67.12, 74.19, 78.83, 82.04, 84.37, 86.13 corresponding to η between 1.89 and 2.11 respectively for the value of $q \in \{50, 100, 150, 200, ..., 400\}$;

(iv) scenario b) with the same values of α^2 as in (iii) corresponding to a dependence parameter $s = 0.99$.

For each set of parameters, (i) to (iv), for $n \in \{50, 75, 100, \ldots, 200\}$, we generate n r.v.'s of size $q \in \{50, 100, 150, \ldots, 400\}$ with $q \geq n$. The procedure is repeated $K = 999$ times independently to obtain Monte-Carlo approximations respectively of the distributions of the penalized Hotelling's T_n^2 statistic (with estimated parameters) and the distribution of the penalizing parameter $\hat{\rho}_n^*$.

The graphics in Figure 3.3 compare the distribution of $\hat{\rho}^*$ for case (i) (independent case, first column) and case (ii) (dependent case, second column of the panel) respectively.
- on the first row: for fixed sample size $n = 50$ and varying $q's$ equal 50, 200 and 400,
- on the second row : for $q = n$ equal to $50, 100, 200$,
- on the last row shows this distribution when $q = 2n$ and n is equal respectively to $50, 100, 200$.

The figures in panel 3.3 show that the dependence structure tends to lead to smaller penalization constants. By comparing the rows, it seems that there is a proportionality between the penalization parameter $\hat{\rho}^*$ and q/n.

In the independent case, it seems to be of the order $2q/n$ up to some factor probably depending on the variance of the eigenvalues of the matrix. Notice that when $q = n$ the center of the distribution is rather stable but with a smaller variance as n grows. In the dependent case, the "optimal" penalization can decrease drastically even if the value of α is fixed but is even more stable (in mean). This can be explained by the fact that we have

$$\alpha^2 = \|S^2 - \sigma^2 I_q\|^2 = \frac{1}{q}\sum_{k=1}^{q}\left(\sigma_k^2 - \sigma^2\right)^2 + \frac{2}{q}\sum_{k,j<k}^{q} cov^2(Z_{1,k}, Z_{1,j}).$$

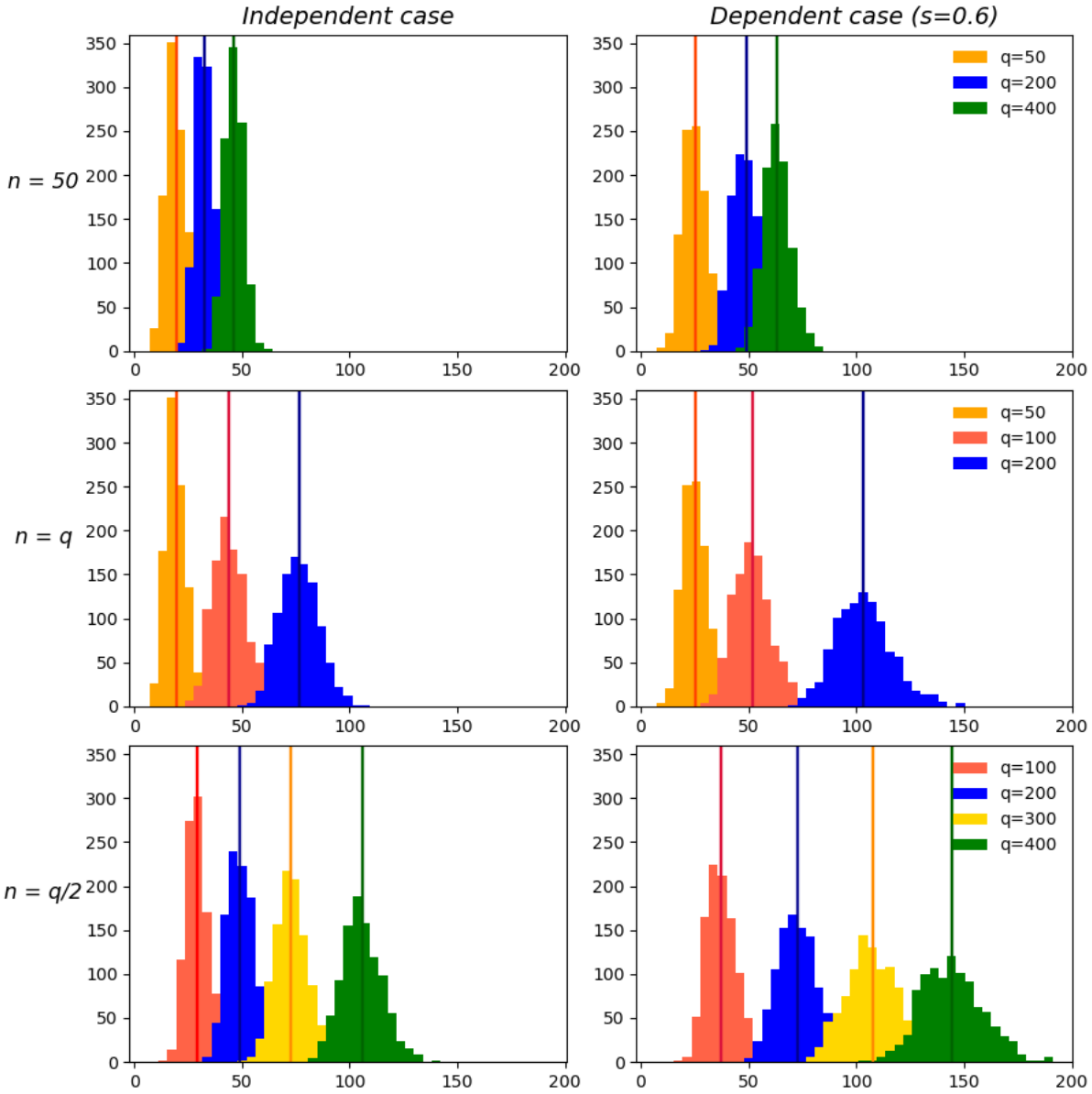


Figure 3.3: **Distributions of $\hat{\rho}^*$, independent (first column, case (i)), dependent with s=0.6 (second column, case (ii)). Vertical lines represent the empirical mean of the corresponding distribution.**

In the independent case, α^2 is essentially the empirical variance of the eigenvalues. But in the dependent case, the covariance terms clearly increase which induces a reduction of the penalizing term since $\rho^* = \frac{\beta^2}{\alpha^2}\sigma^2$.
Now, we focus on the distribution of the optimal penalization when there is a strong dependent component. The graphics panel in Figure 3.4 compares the distribution of $\hat{\rho}^*$ for case (iii) (independent case, first column) and case (iv) (dependent case, second column) respectively on the first row for fixed sample size $n = 50$ and varying $q's$, on the second row for $q = n$ varying in $\{50, 100, 200\}$. Finally, the last row shows this distribution when $q = 2n$ and n varies in $\{50, 100, 200\}$.

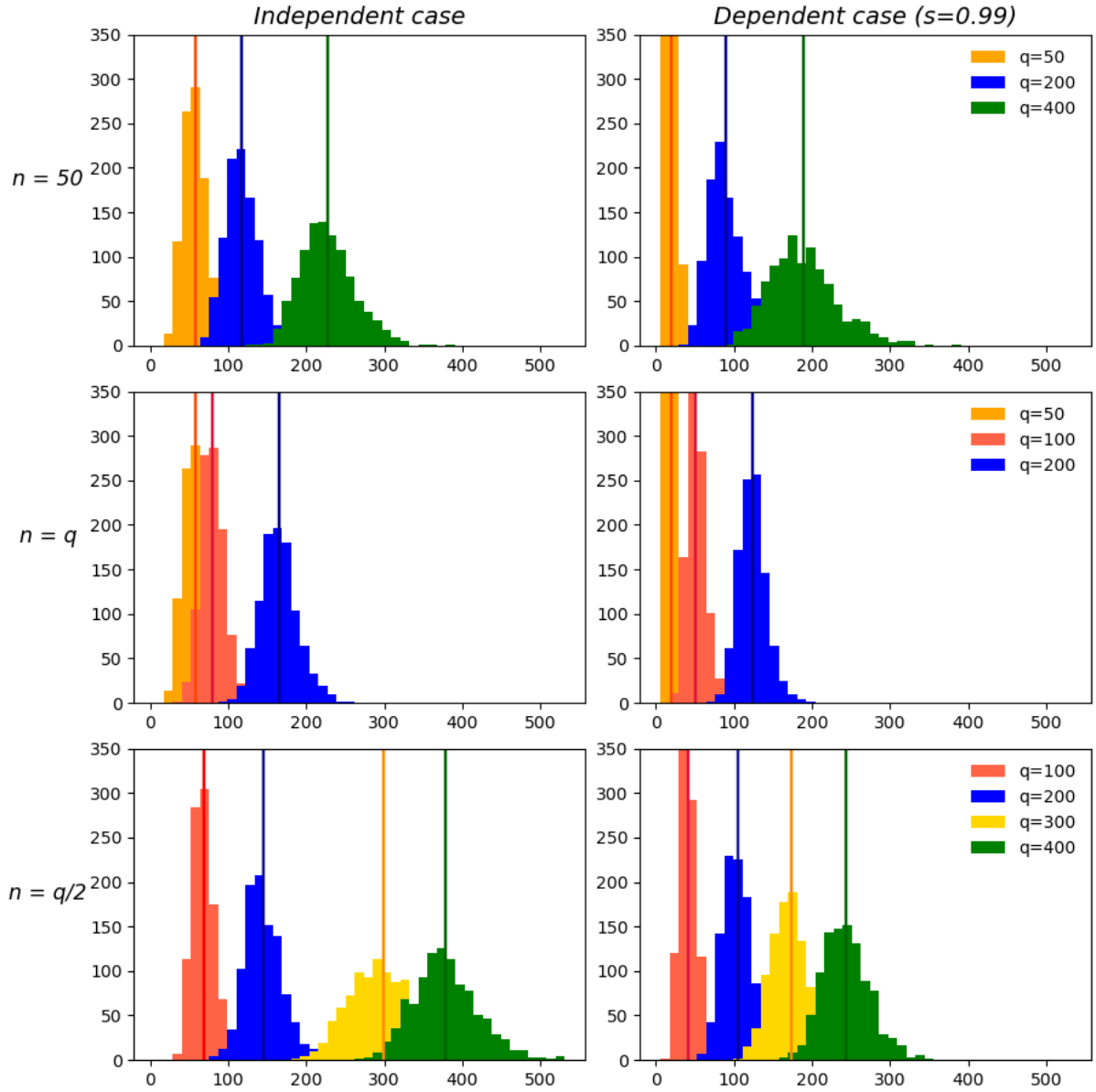


Figure 3.4: **Distributions of $\hat{\rho}^*$, independent (first column, case (iii)), dependent with s=0.99 (second column, case (iv)) with the same α^2. Vertical lines represent the empirical mean of the corresponding distribution.**

Figure 3.4 compares the distribution of the "optimal" estimated penalty for identical values of α^2 (depending on q) for the two scenarios, that is, the left (i.i.d.) and the right column (dependent case) and for different values of q. α^2 is equal respectively to $35.74, 55.63, 67.12, 74.19, 78.83,\ 82.04, 84.37, 86.13$ for the values of q equal to $50, 100, 150, 200, \ldots, 400$. We see that, even for an identical value of α^2, i.e. for a given distance between the true covariance matrix and the diagonal matrix, the distribution of optimal penalty term is systematically more concentrated around smaller values in the dependent case (second column). This conclusion is true for all values of q and n. In other words, the stronger the dependence, the smaller the optimal penalty term.
Recall that, in Figure 3.3, we consider a fixed value $\alpha^2 = 1, 10$ for all values of q. The comparison

of Figures 3.3 and 3.4 shows that when the α^2 term is big, this leads to a smaller penalization term. Furthermore, this penalization becomes smaller when q grows with n. This is quite in contradiction with the practice which suggests using a penalization of the order $2q/n$ as noticed in Figure 3.3. The distance to the homoscedastic framework has thus a very strong impact on the penalty.
The following Figure 3.5 and Figure 3.6 give the histogram of the penalized Hotelling's statistic obtained by $K = 999$ Monte-Carlo simulations, respectively for the independent and dependent case but with the same α^2. We present first the case for $s = 0.6$ (Figure 3.5) and then the case $s = 0.99$ (Figure 3.6).

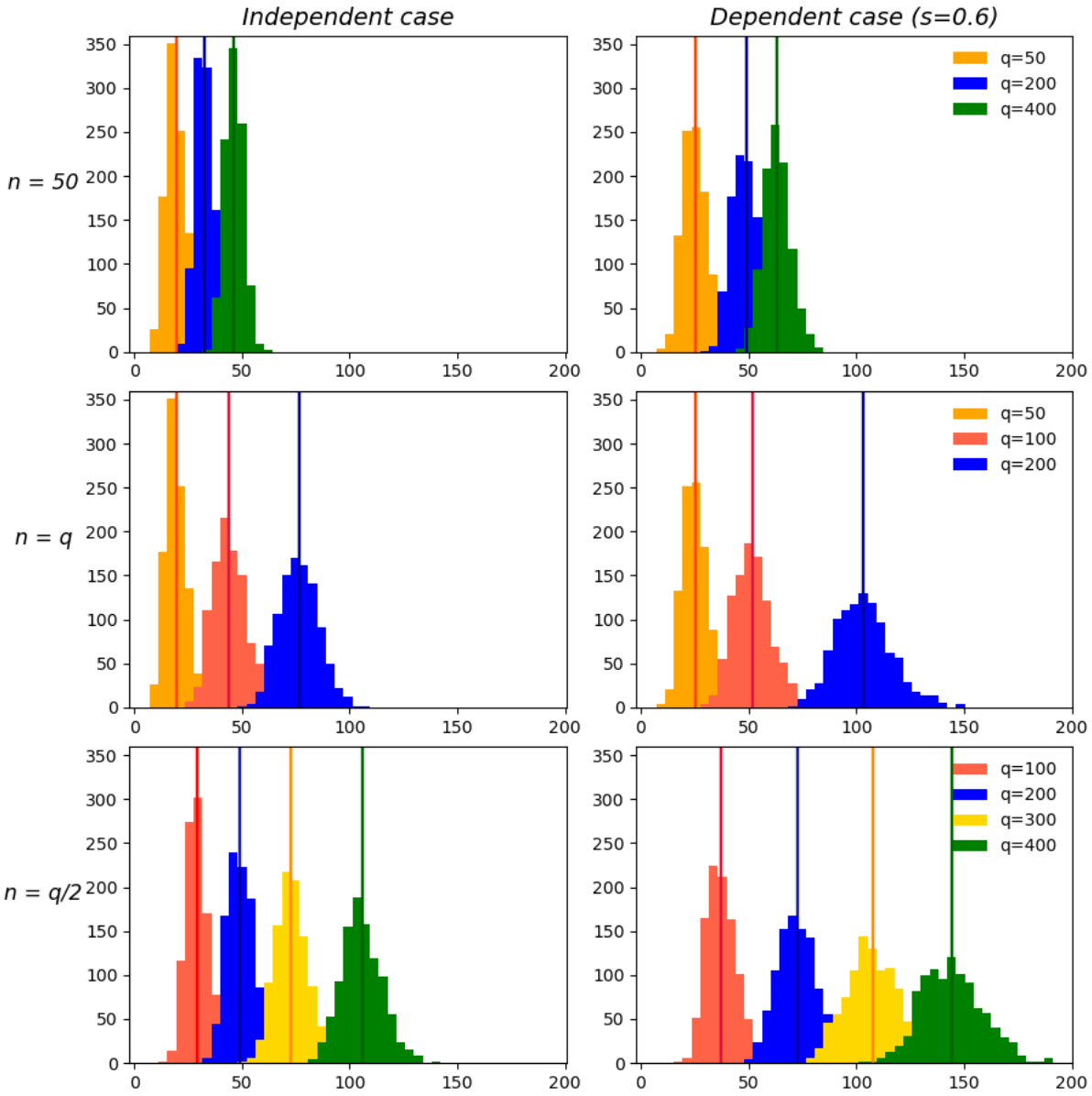


Figure 3.5: **Distributions of $T_n^2(\widehat{\rho}_n^\star, 1)$, the penalized Hotelling's statistic, in independent (first column, case (i)) and dependent with s=0.6 frameworks (second column, case (ii)).**

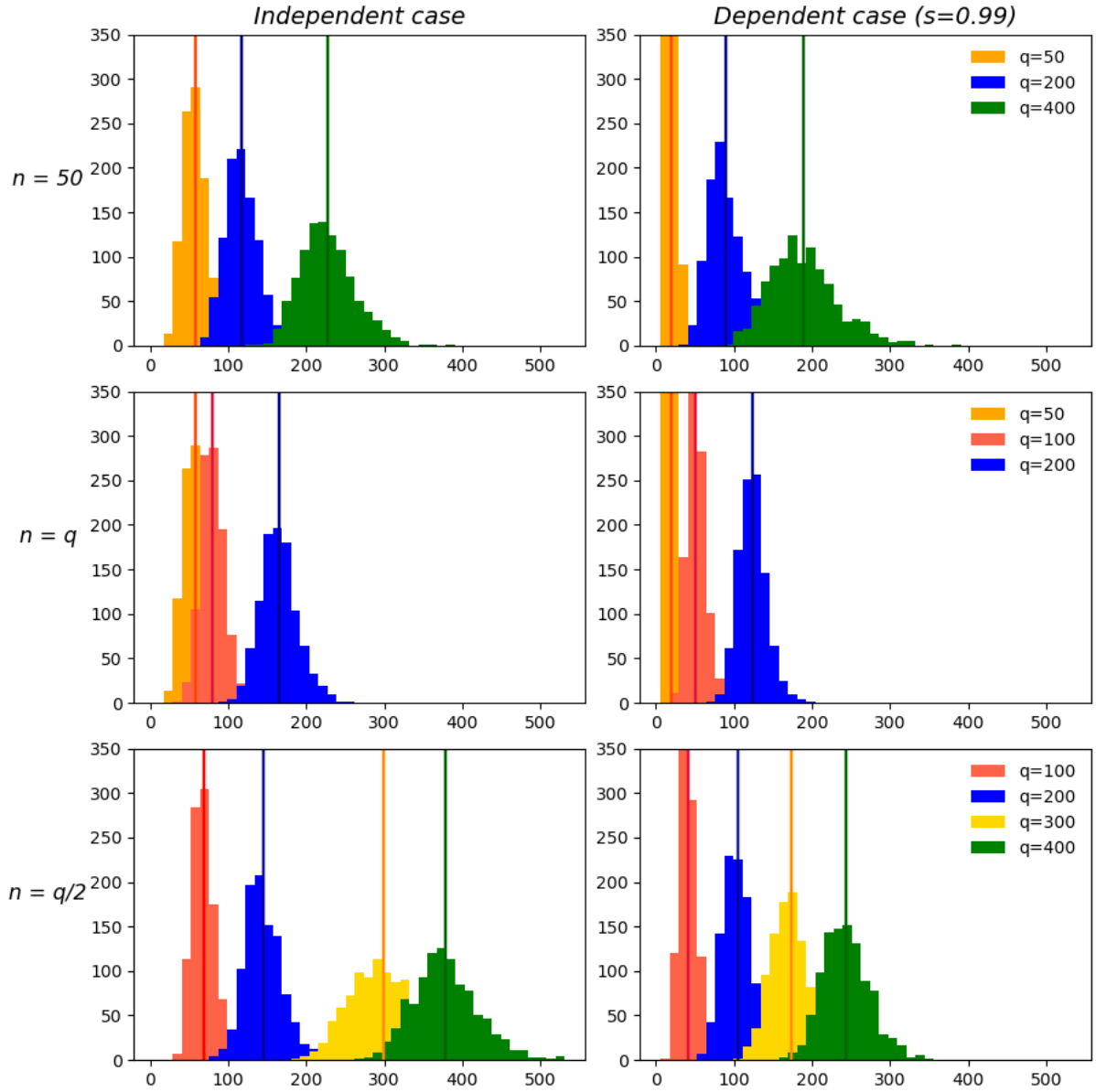


Figure 3.6: **Distributions of $T_n^2(\widehat{\rho}_n^\star, 1)$, the penalized Hotelling's statistic, in independent (first column, case (iii)) and dependent with s=0.99 frameworks (second column, case (iv))**

Compare figures 3.5 and 3.6, focusing first on the first column corresponding to the independent case. We see the importance of the value α (the distance to homoscedasticity) in the distribution. Increasing α^2 tends to lead to a smaller penalization and to a less precise approximation of the covariance matrix yielding a shift of the distribution of the Hotelling's statistic on the right. Comparing the two columns (independent and dependent case), we see that the distributions are centered around quite similar values but tend to be more concentrated in the independent case. Increasing the value of α^2 in figure 3.6 tends to reverse this phenomenon. These figures also emphasize the role of the ratio q/n.
Figures 3.7 and 3.8 show the comparison between the survival function of $T_n^2(\widehat{\rho}_n^*, 1)/(1 + \hat{a}_n^*)$, the penalized Hotelling's statistic reduced by $1 + \hat{a}_n^\star$ compared to the bound obtained in the Theorem 3.3.1. These figures show clearly that the bounds we obtained are too conservative. Curiously the bounds seem to be better when the dependence is very strong.

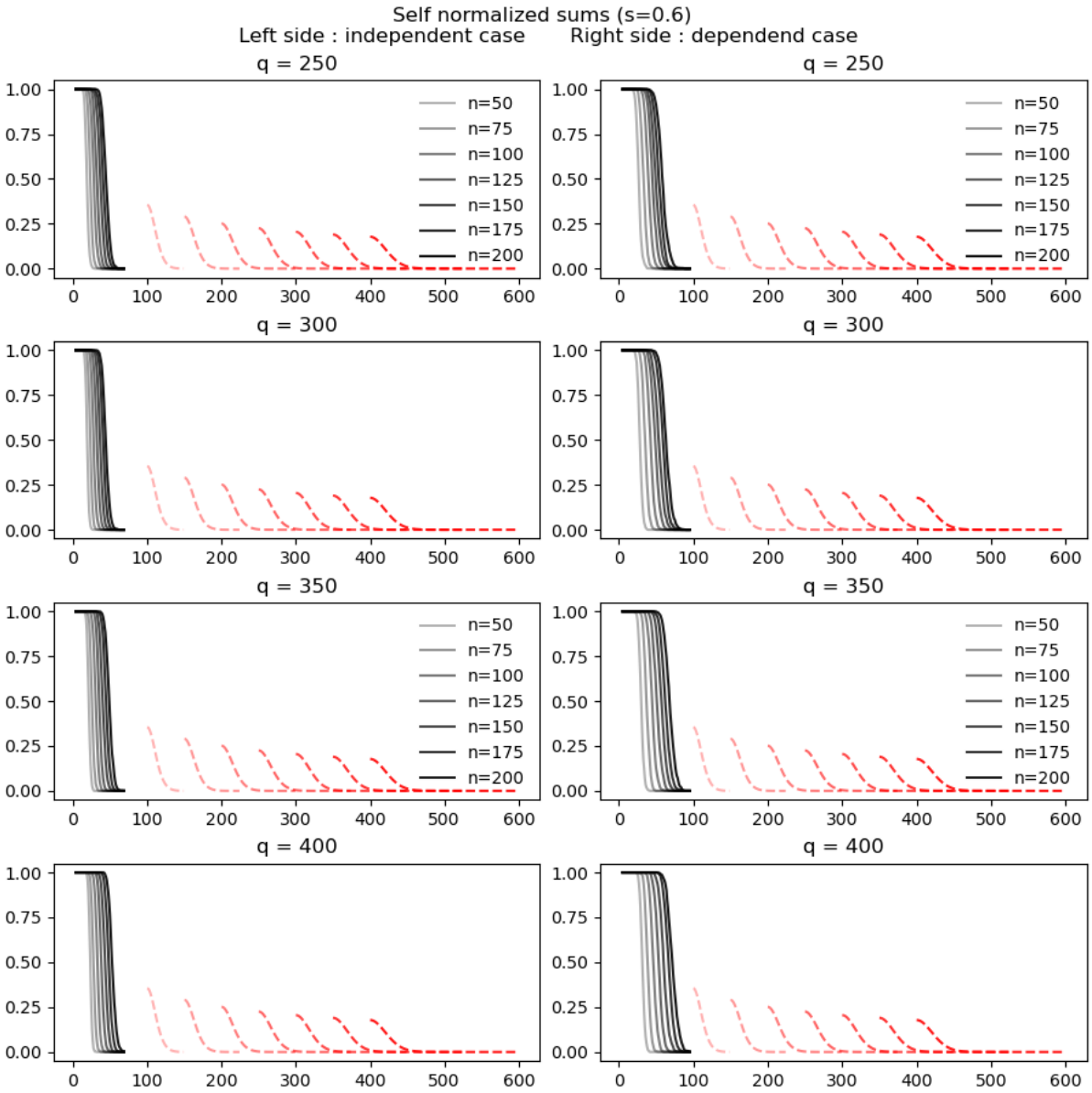


Figure 3.7: **Comparison of the true tail of the penalized Hotelling's statistic and the tail given by the bound for different values of n, q. $s = 0.6$ in the right column. The red dotted lines refer to the bounds for the ordered corresponding n.**

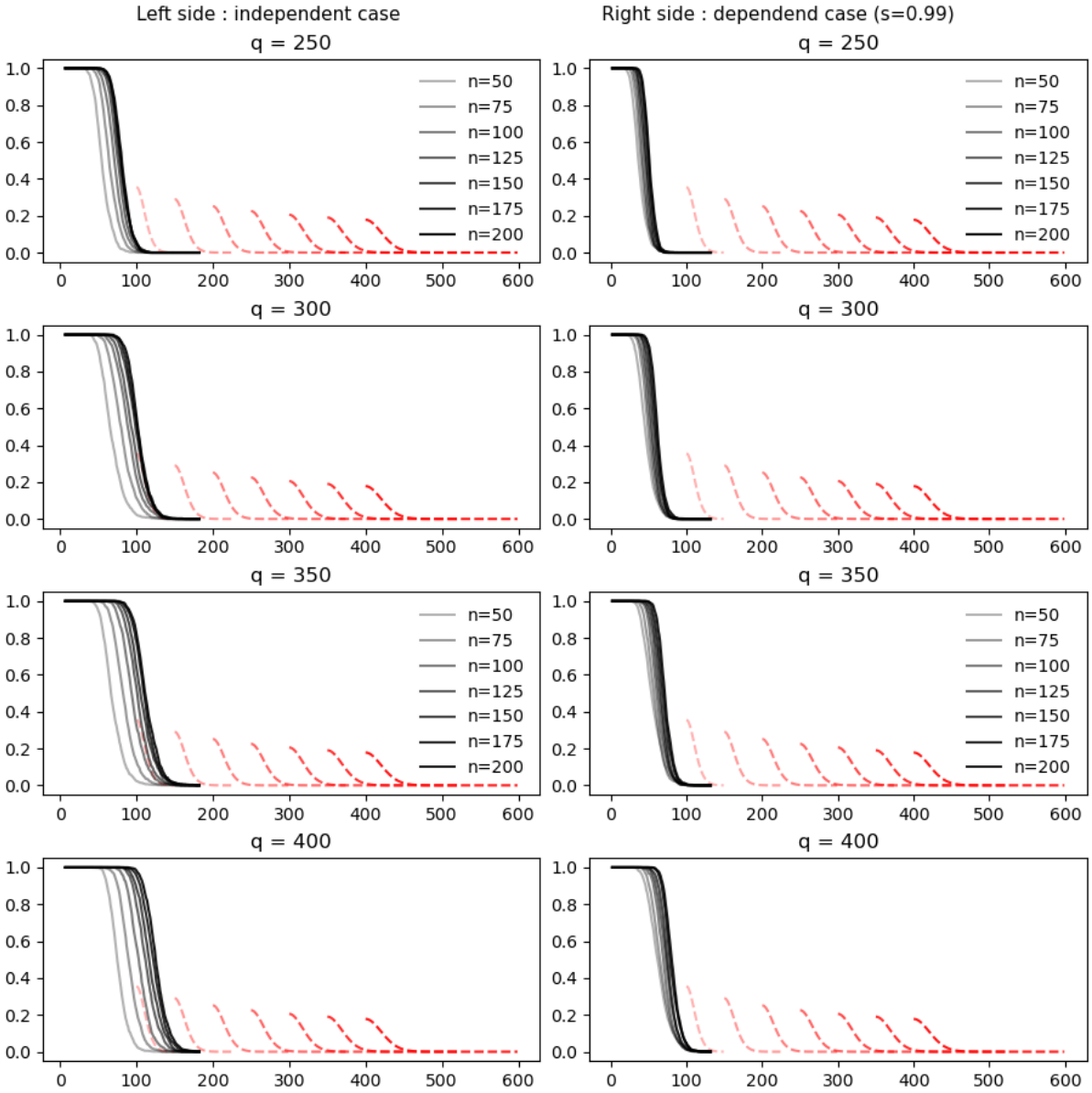


Figure 3.8: **Comparison of the true tail of the penalized Hotelling's statistic and the tail given by the bound for different values of n, q. $s = 0.99$ in the right column. The red dotted lines refer to the bounds for the ordered corresponding n.**

From this simulation study, we conclude that our bounds give some interesting information both on the optimal penalty that one may choose and on the order of the bounds. However, there is still room for improvement.

Appendix

In this appendix, we provide all the proofs of the theorems given in the chapter. In the first part of this section we provide all the proofs of Theorem 3.2.1, 3.2.2, 3.2.3 and 3.3.1 given in the sections 2, 3 and 4. In the second part of the appendix, we detail all the calculations to obtain an explicit constant $C(\epsilon)$ appearing in Theorem 3.3.1 when replacing the true quantities by their empirical estimators. For this, we set some notations that we will consider in the following proofs:
S_n^2 is a symmetric and diagonalizable matrix. Let's denote by O_n an orthogonal matrix in $\mathcal{M}_q(\mathbb{R})$ such that $S_n^2 = O_n' \Lambda_n^2 O_n$ where Λ_n^2 is a diagonal matrix and for any $q > n$

$$\Lambda_n^2 = \begin{pmatrix} \lambda_1 & & & & & \\ & \ddots & & & & \\ & & \lambda_n & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{pmatrix}.$$

Put $\hat{Y}_i = O_n Z_i$ with $\hat{Y}_i = \left(\hat{Y}_{i,1}, ..., \hat{Y}_{i,q}\right)'$. Let $\lambda_1 \geq ... \geq \lambda_q$ denote eigenvalues of S_n^2 and $v_1, ..., v_q$ their associated eigenvectors.

3.4.1 Proof of theorem 3.2.1 and 3.2.2

We first establish a simple inequality for the penalized Hotelling's T_n^2 in the symmetric case, based on previous results by Pinelis (1994) [157]. The idea of the theorem is to use a rotation trick of the Z_i that allows us to return to the "small" dimension case given by Pinelis. This yields a bound given by the survival function of a χ^2 with n degrees of freedom.

Proof of theorem 3.2.1. Note that Vectors $\hat{Y}_i$ remain symmetric in distribution and uncorrelated. It is easy to see that, by construction, the empirical covariance matrix of the $\hat{Y}_1, ..., \hat{Y}_n$ is

$$\frac{1}{n}\sum_{i=1}^{n} \hat{Y}_i \hat{Y}_i' = \frac{1}{n}\sum_{i=1}^{n} O_n Z_i Z_i' O_n' = O_n S_n^2 O_n' = \Lambda_n^2.$$

This implies that, for any vector $\hat{Y}_i$, their coordinates for $j \geq n+1$ are zero. Indeed, for $j \geq n+1$, $n^{-1}\sum_{i=1}^{n} \hat{Y}_{i,j}^2 = 0$, implies in turn that each $\hat{Y}_{i,j} = 0$, for $j = n+1, ..., q$ and $i = 1, ..., n$. Define $\tilde{Y}_i$ the n-dimensional vector version of $\hat{Y}_i$ with these non-zero components, that is to say $\forall j \leq n$, $\tilde{Y}_{i,j} = \hat{Y}_{i,j}$ and their corresponding empirical mean $\bar{\tilde{Y}}_n$ on the collection $\tilde{Y}^{(n)} = \left(\tilde{Y}_i\right)_{1 \leq i \leq n}$.

Thus, for all $\rho_2 > 0$, we have:

$$\begin{aligned} n\bar{Z}_n'\Sigma_n^{-2}(\rho_1,\rho_2)\bar{Z}_n &= n\left(\frac{1}{n}\sum_{i=1}^n \hat{Y}_i'\right)\left(\rho_1 I_q + \rho_2\Lambda_n^2\right)^{-1}\left(\frac{1}{n}\sum_{i=1}^n \hat{Y}_i\right) \\ &= n\sum_{j=1}^n \frac{\left(n^{-1}\sum_{i=1}^n \hat{Y}_{i,j}\right)^2}{\rho_1+\rho_2\lambda_j} \\ &= n\sum_{j=1}^n \frac{\left(n^{-1}\sum_{i=1}^n \hat{Y}_{i,j}\right)^2}{\rho_2\lambda_j}\frac{\rho_2\lambda_j}{\rho_1+\rho_2\lambda_j} \\ &\leq n\sum_{j=1}^n \frac{\left(n^{-1}\sum_{i=1}^n \hat{Y}_{i,j}\right)^2}{\rho_2\lambda_j} \\ &\leq \frac{1}{\rho_2}\sum_{j=1}^n \frac{\left(n^{-1/2}\sum_{i=1}^n \hat{Y}_{i,j}\right)^2}{\lambda_j}. \end{aligned}$$

As $\lambda_j = n^{-1}\sum_{i=1}^n \hat{Y}_{i,j}^2$, we have reduced the problem to the sum of n self normalized sums, which can be seen as Hotelling's T_n^2 of symmetric random variables in $\mathbb{R}^n$. In other words, $n\bar{Z}_n'\Sigma_n^{-2}(\rho_1,\rho_2)\bar{Z}_n \leq \frac{1}{\rho_2} n\bar{\tilde{Y}}_n' S_n^{-2}\left(\tilde{Y}^{(n)}\right)\bar{\tilde{Y}}_n$. Thus, by applying Pinelis' equation (3.1) (1994) [157], we have

$$\forall t>0, \quad \mathbb{P}\left(n\bar{Z}_n'\Sigma_n^{-2}\bar{Z}_n \geq t/\rho_2\right) \leq \frac{2e^3}{9}\bar{F}_n(t)\,.$$

Recall that, if $N_1, ..., N_n$ are independent $N(0,1)$ random variables, then by Lemma 1 of Laurent and Massart (2000) [110], one has, for $u>0$,

$$\mathbb{P}\left(\frac{\sum_{i=1}^n N_i^2 - n}{\sqrt{2n}} \geq \sqrt{2}(\sqrt{u} + \frac{u}{\sqrt{n}})\right) \leq e^{-u}\,.$$

By inverting the polynomial in $\sqrt{u}$, this is a Bernstein type inequality for i.i.d random variables

$$\begin{aligned} \mathbb{P}\left(\frac{\sum_{i=1}^n N_i^2 - n}{\sqrt{2n}} \geq \nu\right) &\leq \exp\left(-\frac{2\nu^2}{\left(1+\sqrt{1+2\sqrt{2}\frac{\nu}{\sqrt{n}}}\right)^2}\right) \\ &\leq \exp\left(-\frac{\nu^2}{2(1+\sqrt{2}\frac{\nu}{\sqrt{n}})}\right). \end{aligned}$$

Il follows that, for $t > n$,

$$\begin{aligned} \bar{F}_n(t) &= \mathbb{P}\left(\frac{\sum_{i=1}^n N_i^2 - n}{\sqrt{2n}} \geq \frac{t-n}{\sqrt{2n}}\right) \\ &\leq \exp\left(-\frac{(t-n)^2}{4t}\right). \end{aligned}$$

□

Proof of theorem 3.2.2.
Recall that : $\bar{Z}_n = \frac{1}{n}\sum_{i=1}^n Z_i$ with $Z_i \in \mathbb{R}^q$. Introduce independent Rademacher r.v.'s ε_i taking the values ± 1 with probability 1/2. Define $\bar{Z}_n^\varepsilon = \frac{1}{n}\sum_{i=1}^n \varepsilon_i Z_i$. Then, in the symmetric case considered here, $\bar{Z}_n$ and $\bar{Z}_n^\varepsilon$ have the same distribution. Now write

$$\begin{aligned} n\bar{Z}_n^{\varepsilon\prime}\Sigma_n^{-2}(\rho_1,\rho_2)\,\bar{Z}_n^\varepsilon &= n\left(\frac{1}{n}\sum_{i=1}^n \varepsilon_i \hat{Y}_i'\right)\left(\rho_1 I_q + \rho_2\Lambda_n^2\right)^{-1}\left(\frac{1}{n}\sum_{i=1}^n \varepsilon_i \hat{Y}_i\right) \quad (3.6)\\ &= \varepsilon' V V'\varepsilon \quad (3.7)\end{aligned}$$

where $\hat{Y} = (\hat{Y}_1,\ldots,\hat{Y}_n)'$, $\varepsilon = (\varepsilon_1,\ldots,\varepsilon_n)'$ and $V = \frac{1}{\sqrt{n}}\hat{Y}\left(\rho_1 I_q + \rho_2\Lambda_n^2\right)^{-1/2}$.

Chasapis and al (2022) [41] obtain an extension of Pinelis' result [157] stating that for smooth functions of quadratic forms, Rademacher variables may be replaced by standard normal variables. More precisely, define the Euclidian norm $\|x\|_2 = \sqrt{\langle x,x\rangle}$ and consider $\xi_1,\ldots,\xi_n$ independent standard Gaussian random variables. Then, for any $t \geq 0$, for any vectors $\nu_1,\ldots,\nu_n$ in $\mathbb{R}^q$, we have

$$\mathbb{P}\left[\|\varepsilon_1\nu_1 + \cdots + \varepsilon_n\nu_n\|_2 \geq t\right] \leq C\mathbb{P}\left[\|\xi_1\nu_1 + \cdots + \xi_n\nu_n\|_2 \geq t\right] \quad \text{with } C = 3824.$$

Since we have

$$\varepsilon_1\nu_1 + \cdots + \varepsilon_n\nu_n = \varepsilon' V$$

where V is the matrix of vectors $\nu_i = (\nu_{i1},\ldots,\nu_{iq})$ corresponding to the rows, we can rewrite

$$\|\varepsilon_1\nu_1 + \cdots + \varepsilon_n\nu_n\|_2^2 = \|\varepsilon' V\|_2^2 = \varepsilon' V V'\varepsilon.$$

It follows that, for any $u > 0$,

$$\mathbb{P}\left[\varepsilon' V V'\varepsilon \geq u\right] \leq C\mathbb{P}\left[\xi' V V'\xi \geq u\right]$$

By conditioning according to $\hat{Y}_i$'s and using equation (3.7), we have, for any $u > 0$ and, for any $\rho_1,\rho_2 > 0$,

$$\begin{aligned}\mathbb{P}\left[n\bar{Z}_n^{\varepsilon\prime}\Sigma_n^{-2}(\rho_1,\rho_2)\,\bar{Z}_n^\varepsilon \geq u\right] &= \mathbb{E}\left[\mathbb{P}\left(\varepsilon' V V'\varepsilon \geq u \mid \hat{Y}_1,\ldots,\hat{Y}_n\right)\right]\\ &\leq C\mathbb{E}\left[\mathbb{P}\left(\xi' V V'\xi \geq u \mid \hat{Y}_1,\ldots,\hat{Y}_n\right)\right].\end{aligned}$$

Moreover, recall from the preceding proof that we have

$$\begin{aligned} n\bar{Z}_n^{\varepsilon\prime}\Sigma_n^{-2}(\rho_1,\rho_2)\,\bar{Z}_n^\varepsilon &= n\left(\frac{1}{n}\sum_{i=1}^n \varepsilon_i \hat{Y}_i'\right)\left(\rho_1 I_q + \rho_2\Lambda_n^2\right)^{-1}\left(\frac{1}{n}\sum_{i=1}^n \varepsilon_i \hat{Y}_i\right)\\ &= n\sum_{j=1}^{\inf(q,n)} \frac{\left(n^{-1}\sum_{i=1}^n \varepsilon_i \hat{Y}_{i,j}\right)^2}{\rho_1 + \rho_2\lambda_j}\\ &= n\sum_{j=1}^{\inf(q,n)} \frac{\left(n^{-1}\sum_{i=1}^n \varepsilon_i \hat{Y}_{i,j}\right)^2}{\lambda_j}\frac{\lambda_j}{\rho_1 + \rho_2\lambda_j}\end{aligned}$$

We obtain

$$\mathbb{P}\left[n\bar{Z}_n^{\varepsilon\prime}\left(\Sigma_n^2(\rho_1,\rho_2)\right)^{-1}\bar{Z}_n^\varepsilon > u\right] \leq C\mathbb{E}\left[\mathbb{P}\left(n\sum_{j=1}^{\inf(q,n)} \frac{\left(n^{-1}\sum_{i=1}^n \xi_i \hat{Y}_{i,j}\right)^2}{\lambda_j}\frac{\lambda_j}{\rho_1 + \rho_2\lambda_j} > u \,\middle|\, \hat{Y}_1,\ldots,\hat{Y}_n\right)\right]. \quad (3.8)$$

Let us work now conditionally to $\hat{Y}_1, \ldots, \hat{Y}_n$. Put $K_j = \sqrt{n}\left(n^{-1}\sum_{i=1}^n \xi_i \hat{Y}_{i,j}\right)/\sqrt{\lambda_j}$ for $j \in \{1, \ldots, \inf(q,n)\}$. Thus for any $j \neq k$

$$\begin{aligned}\mathrm{Cov}\left(K_j, K_k \mid \hat{Y}_1, \ldots, \hat{Y}_n\right) &= \mathrm{Cov}\left(\sqrt{n}\frac{n^{-1}\sum_{i=1}^n \xi_i \hat{Y}_{i,j}}{\sqrt{\lambda_j}}, \sqrt{n}\frac{n^{-1}\sum_{i=1}^n \xi_i \hat{Y}_{i,k}}{\sqrt{\lambda_k}} \;\middle|\; \hat{Y}_1, \ldots, \hat{Y}_n\right) \\ &= \frac{1}{n}\sum_{i=1}^n \frac{\hat{Y}_{i,j}\hat{Y}_{i,k}}{\sqrt{\lambda_j\lambda_k}} = 0.\end{aligned}$$

Since $K = \left(K_1, \ldots, K_{\inf(q,n)}\right)$ is a Gaussian vector (as a linear combination of independent variables) it follows that $K_1^2, \ldots, K_{\inf(q,n)}^2$ are iid $\chi^2(1)$.

Now, consider the vector $b = (b_1, \ldots, b_q)$ with nonnegative components (conditionally to $\hat{Y}_{i,j}$'s) defined by

$$b_j = \frac{\lambda_j}{\rho_1 + \rho_2\lambda_j}.$$

A direct application of Laurent and Massart, lemma [110] to $\sum_{j=1}^{\inf(q,n)} b_j\left(K_j^2 - 1\right)$ gives for any $u > 0$

$$\mathbb{P}\left(\sum_{j=1}^{\inf(q,n)} b_j\left(K_j^2 - 1\right) > 2\|b\|_2\sqrt{u} + 2\|b\|_\infty u\right) \leq \exp(-u).$$

In other words, for any $u > 0$, we have

$$\mathbb{P}\left(\frac{\sum_{j=1}^{\inf(q,n)} b_j K_j^2 - \|b\|_1}{\sqrt{2\|b\|_2^2}} > \sqrt{2}\sqrt{u} + \sqrt{2}\frac{\|b\|_\infty}{\|b\|_2}u\right) \leq \exp(-u) \tag{3.9}$$

Now by combining (3.8) and (3.9) we obtain the following result for the recentered version of our quantity of interest,

$$\begin{aligned}&\mathbb{P}\left(\frac{n\bar{Z}_n^{\varepsilon\prime}\Sigma_n^{-2}(\rho_1,\rho_2)\bar{Z}_n^{\varepsilon} - \|b\|_1}{\sqrt{2\|b\|_2^2}} > \sqrt{2}\sqrt{u} + \sqrt{2}\frac{\|b\|_\infty}{\|b\|_2}u\right) \\ &\leq C\mathbb{E}\left[\mathbb{P}\left(\frac{\sum_{j=1}^{\inf(q,n)} b_j K_j^2 - \|b\|_1}{\sqrt{2\|b\|_2^2}} > \sqrt{2}\sqrt{u} + \sqrt{2}\frac{\|b\|_\infty}{\|b\|_2}u \;\middle|\; \left(\hat{Y}_{1,j}, \ldots, \hat{Y}_{n,j}\right)_{j\in\{1,\ldots,\inf(q,n)\}}\right)\right] \\ &\leq C\exp(-u).\end{aligned}$$

The result of the theorem follows by noticing that $\|b\|_k = \Theta_k(\lambda, \rho_1, \rho_2)$, $k \in \{1, 2, \infty\}$ □

3.4.2 Proof of theorem 3.2.3

A symmetrization lemma adapted to χ^2 distribution

The following lemma ensures that if we have a $\chi^2(k)$ type of control for the tail of a random variable ν, which stochastically dominates some random variable ξ, then we are also able to control the tail of ξ. For large values, this tail is essentially the same as the one of a $\chi^2(k)$ distribution. We use exactly the same ideas as in Panchenko's lemma 1 and corollary 1 (which assumes an exponential control of the tail of the distribution of the variable ν).

Lemma 3.4.1. *Let ν and ξ be two real r.v.'s. For $a \in \mathbb{R}$, put $\Phi_a(x) = \max(x - a; 0)$. Assume that:*

(i) for any $a \in \mathbb{R}$,

$$\mathbb{E}\Phi_a(\xi) \leq \mathbb{E}\Phi_a(\nu)$$

(ii) there exists k and constants $\mathsf{C}_1 > 0, c_1 > 0$, such that for any $t > 0$

$$\mathbb{P}(\nu \geq t) \leq \mathsf{C}_1 \bar{F}_k(c_1 t)$$

then, for $t > 2k/c_1$, we have

$$\mathbb{P}(\xi \geq t) \leq \mathsf{C}_1 \left(\frac{c_1 t - k}{2}\right)^{\frac{k}{2}} \frac{e^{-\frac{c_1 t - k}{2}}}{\Gamma\left(\frac{k}{2}+1\right)}$$

and, for $t > k/c_1$, we also get

$$\mathbb{P}(\xi \geq t) \leq \mathsf{C}_1 \bar{F}_{k+2}(c_1 t - k).$$

Proof of lemma 3.4.1 . We follow the lines of the proof of Panchenko's lemma, with a function Φ_a with $a = t - \frac{k}{c_1}$ given by $\Phi(x) = \max(x - t + k/c_1; 0)$, for $t > k/c_1$. Remark that $\Phi(x)$ is convex, nondecreasing and that $\Phi(0) = 0$ and $\Phi(t) = k/c_1$. We thus have by Markov's inequality

$$\begin{aligned}
\mathbb{P}(\xi \geq t) &\leq \frac{E\Phi(\xi)}{\Phi(t)} \leq \frac{E\Phi(\nu)}{\Phi(t)} \\
&\leq \frac{1}{\Phi(t)} \left(\Phi(0) + \int_{t-k/c_1}^{+\infty} \Phi'(x)\, \mathbb{P}(\nu \geq x)\, dx\right) \\
&\leq \mathsf{C}_1 \frac{c_1}{k} \int_{t-k/c_1}^{+\infty} \overline{F}_k(c_1 x)\, dx.
\end{aligned}$$

By integration by parts, we get

$$\int_{t-k/c_1}^{+\infty} \bar{F}_k(c_1 x)\, dx = \int_{t-k/c_1}^{+\infty} c_1 x f_k(c_1 x)\, dx - (t - k/c_1) \int_{t-k/c_1}^{+\infty} c_1 f_k(c_1 x)\, dx.$$

Recall that

$$f_k(u) = \frac{1}{2^{k/2}\Gamma(\frac{k}{2})} u^{\frac{k}{2}-1} \exp(-\frac{u}{2}),$$

we thus have

$$\begin{aligned}
\frac{c_1}{k} \int_{t-k/c_1}^{+\infty} c_1 x f_k(c_1 x)\, dx &= \frac{c_1}{2^{k/2+1}\frac{k}{2}\Gamma(\frac{k}{2})} \int_{t-k/c_1}^{+\infty} (c_1 x)^{\frac{k+2}{2}-1} \exp(-\frac{c_1 x}{2}) dx \\
&= \overline{F}_{k+2}(c_1 t - k).
\end{aligned}$$

It follows by straightforward calculations that, for $t > k/c_1$,

$$\mathbb{P}(\xi \geq t) \leq \mathsf{C}_1 \left(\overline{F}_{k+2}(c_1 t - k) - \frac{c_1 t - k}{k} \overline{F}_k(c_1 t - k)\right).$$

Using the recurrence relation 26.4.8 of Abramovitch and Stegun ([1], page 941), for $u \geq 2k$,

$$\begin{aligned}
\mathsf{C}_1 \left(\overline{F}_{k+2}(u-k) - \tfrac{u-k}{k}\overline{F}_k(u-k)\right) &\leq \mathsf{C}_1 \left(\overline{F}_{k+2}(u-k) - \overline{F}_k(u-k)\right) \\
&\leq \left(\tfrac{(u-k)}{2}\right)^{k/2} \tfrac{\mathsf{C}_1 e^{-\frac{(u-k)}{2}}}{\Gamma\left(\frac{k}{2}+1\right)}.
\end{aligned}$$

We get with $u = c_1 t$, for $t \geq 2k/c_1$,

$$\mathbb{P}(\xi \geq t) \leq \left(\frac{(c_1 t - k)}{2}\right)^{k/2} \frac{\mathsf{C}_1 e^{-\frac{(c_1 t - k)}{2}}}{\Gamma\left(\frac{k}{2}+1\right)}.$$

Moreover, for $t > k/c_1$ we have $\mathbb{P}(\xi \geq t) \leq \mathsf{C}_1 \left(\overline{F}_{k+2}(c_1 t - k)\right)$. Notice that we only lose 2 degrees of freedom in this case. It will not be important if k is large, typically of the order of n in our case.

□

Extension of Panchenko symmetrization lemma (see [152] Corollary 1, p. 2069)

Let $\mathcal{J}_q = \{u \in \mathbb{R}^q, \|u\|_2 = 1\}$ be the unit circle of $\mathbb{R}^q$. Let $X^{(n)} = (X_i)_{1\leq i\leq n}$ be an independent copy of $Z^{(n)} = (Z_i)_{1\leq i\leq n}$.
Since $q > n$, the matrix $S_n^2\left(Z^{(n)} - X^{(n)}\right) = \frac{1}{n}\sum_{i=1}^n (Z_i - X_i)(Z_i - X_i)'$ is not invertible. We derive from $S_n^2\left(Z^{(n)} - X^{(n)}\right)$ the corresponding penalized empirical covariance matrix

$$\tilde{\Sigma}_n^2 = 2\rho_1 I_q + \rho_2 S_n^2\left(Z^{(n)} - X^{(n)}\right)$$

It is easy to see that

$$\mathbb{E}\left(S_n^2\left(Z^{(n)} - X^{(n)}\right)\right) = 2S^2 \text{ and } \mathbb{E}\left(S_n^2\left(Z^{(n)} - X^{(n)}\right) \mid Z^{(n)}\right) = S_n^2 + S^2.$$

Since $\tilde{\Sigma}_n^2 = \tilde{\rho}_1 I_q + \tilde{\rho}_2 S_n^2\left(Z^{(n)} - X^{(n)}\right)$, we get that

$$\mathbb{E}\left(\tilde{\Sigma}_n^2 \mid Z^{(n)}\right) = \tilde{\rho}_1 I_q + \tilde{\rho}_2\left(S_n^2 + S^2\right) = 2\rho_1 I_q + \rho_2\left(S_n^2 + S^2\right).$$

As a consequence, define

$$\begin{aligned}
\tilde{\beta}^2 &= \mathbb{E}\left(\left\|S_n^2\left(Z^{(n)} - X^{(n)}\right) - 2S^2\right\|^2\right) \\
&= \mathbb{E}\left(\left\|S_n^2\left(Z^{(n)}\right) - S^2\right\|^2\right) + \mathbb{E}\left(\left\|S_n^2\left(X^{(n)}\right) - S^2\right\|^2\right) \\
&= 2\beta^2.
\end{aligned}$$

Similarly, put

$$\tilde{\alpha}^2 = 2\alpha^2;\ \tilde{\delta} = 2\delta^2 \text{ and } \tilde{\sigma}^2 = \left\langle 2S^2, I_n\right\rangle = 2\sigma^2$$

then we have

$$\tilde{\rho}_1 = \frac{\tilde{\alpha}^2}{\tilde{\delta}^2}\tilde{\sigma}^2 = 2\frac{\alpha^2}{\delta^2}\sigma^2 = 2\rho_1$$

and

$$\tilde{\rho}_2 = \frac{\tilde{\beta}^2}{\tilde{\delta}^2} = \frac{\beta^2}{\delta^2} = \rho_2$$

It thus follows with this natural choice of $\widetilde{\rho}_1$ and $\widetilde{\rho}_2$ that we have

$$\mathbb{E}\left(\tilde{\Sigma}_n^2 \mid Z^{(n)}\right) = \Sigma_n^2 + \Sigma^2$$

$$\mathbb{E}\left(\tilde{\Sigma}_n^2\right) = 2(\rho_1 I_q + \rho_2 S^2) = 2\Sigma^2$$

The following lemma and its proof is an extension of corollary 1 of Panchenko (2003) (see [152]) with some adaptations to the multidimensional χ^2 case. See also Bertail et al. (2008) [14] for the non penalized version of this result for $q < n$.

Lemma 3.4.2. *If there exists $k \in \mathbb{N}^*$, $C_2 > 0$ and $c_2 > 0$ such that, for all $t \geq 0$,*

$$\mathbb{P}\left(\sup_{u\in\mathcal{J}_q}\left(\frac{\sqrt{n}u'\left(\bar{Z}_n - \bar{X}_n\right)}{\sqrt{u'\tilde{\Sigma}_n^2 u}}\right) \geq \sqrt{t}\right) \leq C_2\bar{F}_k(c_2 t),$$

then, for all $t \geq 2k/c_2$,

$$\mathbb{P}\left(\sup_{u\in\mathcal{J}_q}\left(\frac{\sqrt{n}u'\bar{Z}_n}{\sqrt{u'\left(\Sigma_n^2+\Sigma^2\right)u}}\right)\geq\sqrt{t}\right)\leq \mathsf{C}_2\left(\frac{(c_2t-k)}{2}\right)^{k/2}\frac{e^{-\frac{(c_2t-k)}{2}}}{\Gamma\left(\frac{k}{2}+1\right)}$$

and, for all $t \geq k/c_2$,

$$\mathbb{P}\left(\sup_{u\in\mathcal{J}_q}\left(\frac{\sqrt{n}u'\bar{Z}_n}{\sqrt{u'\left(\Sigma_n^2+\Sigma^2\right)u}}\right)\geq\sqrt{t}\right)\leq \mathsf{C}_2\overline{F}_{k+2}\left(c_2t-k\right)$$

Proof of Lemma 3.4.2 . Denote

$$A_n\left(Z^{(n)}\right)=n\operatorname*{supsup}_{u\in\mathcal{J}_q\ b>0}\left\{\mathbb{E}\left[4b\left(u'\left(\bar{Z}_n-\bar{X}_n\right)-bu'\tilde{\Sigma}_n^2u\right)\mid Z^{(n)}\right]\right\}$$

and

$$C_n\left(Z^{(n)},X^{(n)}\right)=n\operatorname*{supsup}_{u\in\mathcal{J}_q\ b>0}\left\{4b\left(u'\left(\bar{Z}_n-\bar{X}_n\right)-bu'\tilde{\Sigma}_n^2u\right)\right\}$$

We have by Jensen's inequality, that for any convex function ϕ

$$\phi\left(A_n\left(Z^{(n)}\right)\right)\leq\mathbb{E}\left[\phi\left(C_n\left(Z^{(n)},X^{(n)}\right)\right)\mid Z^{(n)}\right] \tag{3.10}$$

Finally, we can rewrite $A_n\left(Z^{(n)}\right)$ and $C_n\left(Z^{(n)},X^{(n)}\right)$ in an explicit form of self-normalized sums by maximizing according to b, the two expressions above, which leads to

$$\begin{aligned}A_n\left(Z^{(n)}\right)&=\sup_{u\in\mathcal{J}_q}\left\{\left(\frac{\sqrt{n}u'\bar{Z}_n}{\sqrt{\tilde{\rho}_1+\tilde{\rho}_2u'\left(S_n^2+S^2\right)u}}\right)^2\right\}\\&=\sup_{u\in\mathcal{J}_q}\left\{\left(\frac{\sqrt{n}u'\bar{Z}_n}{\sqrt{u'\Sigma_n^2u+u'\Sigma^2u}}\right)^2\right\}\end{aligned}$$

Similarly, we have

$$C_n\left(Z^{(n)},X^{(n)}\right)=\sup_{u\in\mathcal{J}_q}\left\{\left(\frac{\sqrt{n}u'\left(\bar{Z}_n-\bar{X}_n\right)}{\sqrt{u'\tilde{\Sigma}_n^2u}}\right)^2\right\}$$

Now we conclude by applying lemma 1 to the inequality (3.10) with these expressions of $A_n\left(Z^{(n)}\right)$ and $C_n\left(Z^{(n)},X^{(n)}\right)$ with $\mathsf{C}_2=\mathsf{C}_1$ and $c_2=c_1$.

□

Proof of theorem 3.2.3 . We now control the Hotelling's T_n^2 in the general case, by cutting its distribution tail into two parts. The first part allows us to get back to the expression above

$$\sup_{u\in\mathcal{J}_q}\left\{\left(\frac{\sqrt{n}u'\bar{Z}_n}{\sqrt{u'\Sigma_n^2u+u'\Sigma^2u}}\right)^2\right\}$$

controlled by Lemma 2. The second term is controlled by the largest eigenvalue of S^2.

Let

$$B_n=\sup_{u\in\mathcal{J}_q}\left\{\frac{u'\bar{Z}_n}{\sqrt{u'\Sigma_n^2u}}\right\}.$$

Notice that by construction we have, for any $t > 0$, (and particularly for any $t > 2n$)

$$\left\{n\bar{Z}_n'\Sigma_n^{-2}\bar{Z}_n \geq t\right\} = \left\{n^{1/2}B_n \geq \sqrt{t}\right\}.$$

To transform the penalized self-normalised sum from the expression $n\bar{Z}_n'\left(\Sigma_n^2\right)^{-1}\bar{Z}_n$ to its "pseudo" version with the wrong normalization, $\sup_{u\in\mathcal{J}_q}\left\{\left(\frac{\sqrt{n}u'\bar{Z}_n}{\sqrt{u'\Sigma_n^2u+u'\Sigma^2u}}\right)^2\right\}$, let us introduce D_n defined by

$$D_n = \sup_{u\in\mathcal{J}_q}\left\{\sqrt{1+\frac{u'\Sigma^2u}{u'\Sigma_n^2u}}\right\} = \sup_{u\in\mathcal{J}_q}\left\{\sqrt{1+\frac{u'\left(\rho_1I_q+\rho_2S^2\right)u}{u'\left(\rho_1I_q+\rho_2S_n^2\right)u}}\right\}.$$

First, notice that we have

$$\begin{aligned}\sqrt{n}\frac{B_n}{D_n} &= \sup_{u\in\mathcal{J}_q}\left\{\frac{u'\bar{Z}_n}{\sqrt{u'\Sigma_n^2u}}\right\}\inf_{u\in\mathcal{J}_q}\left\{\left(\sqrt{1+\frac{u'\Sigma^2u}{u'\Sigma_n^2u}}\right)^{-1}\right\}\\ &\leq \sup_{u\in\mathcal{J}_q}\left(\frac{u'\bar{Z}_n}{\sqrt{u'\Sigma_n^2u}}\left(\sqrt{1+\frac{u'\Sigma^2u}{u'\Sigma_n^2u}}\right)^{-1}\right)\\ &\leq \sqrt{\sup_{u\in\mathcal{J}_q}\left\{\left(\frac{\sqrt{n}u'\bar{Z}_n}{\sqrt{u'\Sigma_n^2u+u'\Sigma^2u}}\right)^2\right\}},\end{aligned} \tag{3.11}$$

for which we have an exponential bound by Lemma 3.4.2 and theorem 3.2.1.
Thus by splitting the probability according to the event $\{D_n^2 \geq 1+a\}$, for $a > 1$ and, for any $t > 2n$, we have

$$\begin{aligned}\mathbb{P}\left(n\bar{Z}_n'\Sigma_n^{-2}\bar{Z}_n \geq t\right) &\leq \mathbb{P}\left(B_n \geq \sqrt{\frac{t}{n}},\ D_n \leq \sqrt{1+a}\right) + \mathbb{P}\left(D_n \geq \sqrt{1+a}\right)\\ &\leq \mathbb{P}\left(\frac{B_n}{D_n} \geq \sqrt{\frac{t}{n\left(1+a\right)}}\right) + \mathbb{P}\left(D_n \geq \sqrt{1+a}\right).\end{aligned} \tag{3.12}$$

So now, it remains to treat the second term in the right-hand side of inequality (3.12). Notice that we have, for $a > 1$,

$$\begin{aligned}\left\{D_n \geq \sqrt{1+a}\right\} &= \left\{\sup_{u\in\mathcal{J}_q}\left(\frac{u'\Sigma^2u}{u'\Sigma_n^2u}\right) \geq a\right\}\\ &= \left\{\inf_{u\in\mathcal{J}_q}\left(\frac{u'\Sigma_n^2u}{u'\Sigma^2u}\right) \leq \frac{1}{a}\right\}.\end{aligned}$$

First, if $S^2 = \sigma^2I_q$ is diagonal, then we have

$$u'\Sigma^2u = u'(\rho_1I_q+\rho_2\sigma^2I_q)u = \rho_1+\rho_2\sigma^2.$$

Since

$$\inf_{u\in\mathcal{J}_q}\left(u'\Sigma_n^2u\right) = \inf_{u\in\mathcal{J}_q}\left(u'(\rho_1I_q+\rho_2S_n^2)\,u\right) = \rho_1,$$

if we choose a such that $a > (\rho_1+\rho_2\sigma^2)/\rho_1$, then we have

$$\mathbb{P}\left\{D_n \geq \sqrt{1+a}\right\} \leq \mathbb{P}\left(\frac{\inf_{u\in\mathcal{J}_q}\left(u'\Sigma_n^2u\right)}{\rho_1+\rho_2\sigma^2} \leq \frac{1}{a}\right) = 0.$$

Remark that, in this case, we have $\rho_1^* = \sigma^2$ and $\rho_2^* = 0$ and it follows that the inequality is true for any $a > 1$. Notice that the proximity between S^2 and σ^2I_q is precisely controlled by the term

$\alpha^2 = \left\| S^2 - \sigma^2 I_q \right\|$.

Now consider the general case. First, notice that

$$\begin{aligned}
\inf_{u\in\mathcal{J}_q}\left(\frac{u'\Sigma_n^2 u}{u'\Sigma^2 u}\right) &= \inf_{u\in\mathcal{J}_q}\left(u'\Sigma^{-1}\Sigma_n^2\Sigma^{-1}u\right)\\
&= \inf_{u\in\mathcal{J}_q}\left(\frac{u'\Sigma^{-1}}{\left\|\Sigma^{-1}u\right\|_2}\Sigma_n^2\frac{\Sigma^{-1}u}{\left\|\Sigma^{-1}u\right\|_2}\left\|\Sigma^{-1}u\right\|_2^2\right)\\
&\geq \inf_{v\in\mathcal{J}_q}\left(v'\Sigma_n^2 v\right)\times\inf_{u\in\mathcal{J}_q}\left(u'\Sigma^{-2}u\right), \quad \text{with } v=\frac{\Sigma^{-1}u}{\left\|\Sigma^{-1}u\right\|_2}\\
&\geq \rho_1\mu_1(\Sigma^{-2}) = \frac{\rho_1}{\mu_q(\Sigma^2)}.
\end{aligned}$$

Now, using the optimal values ρ_1^* and ρ_2^*, we have the decomposition

$$\Sigma^2\left(\rho_1^*,\rho_2^*\right) = \rho_1^* I_q + \rho_2^* S^2.$$

It follows that we get

$$\mu_q(\Sigma^2\left(\rho_1^*,\rho_2^*\right)) = \rho_1^* + \rho_2^*\mu_q(S^2)$$

and

$$\inf_{u\in\mathcal{J}_q}\left(\frac{u'\Sigma_n^2\left(\rho_1^*,\rho_2^*\right)u}{u'\Sigma^2\left(\rho_1^*,\rho_2^*\right)u}\right) \geq \frac{\rho_1^*}{\rho_1^*+\rho_2^*\mu_q(S^2)}.$$

It follows that if we choose a such that

$$\frac{1}{a} < \frac{1}{1+\frac{\mu_q(S^2)}{\rho^*}}$$

and, since $a^* = 1+\frac{K_3}{\rho^*} > 1+\frac{\mu_q\left(S^2\right)}{\rho^*}$ by the assumption (A_3), then, if $a\geq a^*$, we get

$$\mathbb{P}\left(D_n \geq \sqrt{1+a}\right) = 0. \tag{3.13}$$

As a consequence, we obtain an exponential inequality for any value $a\geq a^*$. Combining (3.12) and (3.13), we get, for any $a\geq a^*$,

$$\forall t > 2n, \quad \mathbb{P}\left(n\bar{Z}_n'\Sigma_n^{-2}\bar{Z}_n \geq t\left(1+a\right)\right) \leq \mathbb{P}\left(\sqrt{n}\frac{B_n}{D_n}\geq\sqrt{t}\right). \tag{3.14}$$

Let $X^{(n)} = (X_i)_{1\leq i\leq n}$ be an independent copy of $Z^{(n)} = (Z_i)_{1\leq i\leq n}$. Applying theorem 3.2.1 to $(Z_i - X_i)_{1\leq i\leq n}$ which is symmetric, we obtain

$$\mathbb{P}\left(\sup_{u\in\mathcal{J}_q}\left(\frac{\sqrt{n}u'\left(\bar{Z}_n-\bar{X}_n\right)}{\sqrt{u'\tilde{\Sigma}_n^2 u}}\right)\geq\sqrt{t}\right) \leq \frac{2e^3}{9}\bar{F}_n\left(t\right),$$

Thus, applying the lemma 3.4.2 to the inequality above implies that, for all $t\geq 2n$,

$$\mathbb{P}\left(\sup_{u\in\mathcal{J}_q}\left(\frac{\sqrt{n}u'\bar{Z}_n}{\sqrt{u'\left(\Sigma_n^2+\Sigma^2\right)u}}\right)\geq\sqrt{t}\right) \leq \frac{2e^3}{9}\left(\frac{(t-n)}{2}\right)^{n/2}\frac{e^{-\frac{(t-n)}{2}}}{\Gamma\left(\frac{n}{2}+1\right)}. \tag{3.15}$$

Finally by combining expressions (3.11), (3.14) and (3.15), the result holds. □

3.4.3 Proof of Theorem 3.3.1

The following lemmas will allow us to control explicitly the deviation $\mathbb{P}\left[\left|\frac{1}{\widehat{\rho}_n^*}-\frac{1}{\rho^*}\right|>\epsilon\right]$ for small positive values of ϵ.

Lemma 3.4.3. (Inversion) *Let $w>0$, and consider $(W_n)_{n\geq 1}$ a sequence of positive random variables. Assume that there exists a nonnegative constant C_3, such that $\forall\epsilon>0,\exists N>0,\forall n>N$,*

$$\mathbb{P}\left(|W_n-w|>\epsilon\right)\leq\frac{\mathsf{C}_3}{n}\frac{1}{\epsilon^2}.$$

Then there exists a function $\mathsf{C}_{3;1/w}$ nonnegative, such that $\forall\epsilon>0,\forall n>N$

$$\mathbb{P}\left(\left|\frac{1}{W_n}-\frac{1}{w}\right|>\epsilon\right)\quad\leq\quad\frac{\mathsf{C}_{3;1/w}\left(\epsilon\right)}{n\epsilon^2},$$

where $\mathsf{C}_{3;1/w}\left(\epsilon\right)=\frac{C_3}{w^4}\left(1+\left(w\epsilon\right)^{2/5}\right)^5$.

Proof of Lemma 3.4.3 . Since $w>0$, we have

$$\mathbb{P}\left(\left|\frac{1}{W_n}-\frac{1}{w}\right|>\frac{\epsilon}{w}\right)=\mathbb{P}\left(\left|\frac{w}{W_n}-1\right|>\epsilon\right)$$

Now, $\forall\eta\in\left]0,w\right[$ we get

$$\begin{aligned}\mathbb{P}\left(\left|\frac{1}{W_n}-\frac{1}{w}\right|>\frac{\epsilon}{w}\right)&\leq\mathbb{P}\left(\left|\frac{w}{W_n}-1\right|>\epsilon,\ |W_n-w|\leq\eta\right)+\mathbb{P}\left(|W_n-w|>\eta\right)\\&\leq\text{(I)}+\text{(II)}.\end{aligned}$$

On the interval $[w-\eta;w+\eta]$, $f:x\mapsto\frac{w}{x}$ is Lipschitz with

$$\forall x\in\left[w-\eta;w+\eta\right],\ \left|f'\left(x\right)\right|\leq\frac{w}{\left(w-\eta\right)^2},$$

thus we obtain

$$\forall W_n\in\left[w-\eta;w+\eta\right],\ \left|\frac{w}{W_n}-1\right|\leq\frac{w}{\left(w-\eta\right)^2}\left|W_n-w\right|.$$

$$\forall\eta\in\left]0;w\right[,\ (I)\leq\mathbb{P}\left(\frac{w}{\left(w-\eta\right)^2}\left|W_n-w\right|>\epsilon\right)\leq\frac{\mathsf{C}_3}{n}\times\frac{w^2}{\epsilon^2\left(w-\eta\right)^4}$$

and since

$$\forall\eta\in\left]0;w\right[,\ (II)\leq\frac{\mathsf{C}_3}{n}\times\frac{1}{\eta^2},$$

it follows that

$$\begin{aligned}\mathbb{P}\left(\left|\frac{1}{W_n}-\frac{1}{w}\right|>\frac{\epsilon}{w}\right)&\leq\frac{\mathsf{C}_3}{n}\times\frac{w^2}{\epsilon^2\left(w-\eta\right)^4}+\frac{\mathsf{C}_3}{n}\times\frac{1}{\eta^2}\\&\leq\frac{\mathsf{C}_3}{n}\min_{\eta\in\left]0;w\right[}\left\{\frac{w^2}{\epsilon^2\left(1-\frac{\eta}{w}\right)^4w^4}+\frac{1}{w^2\left(\frac{\eta}{w}\right)^2}\right\}\\&\overset{\alpha=\frac{\eta}{w}}{\leq}\frac{\mathsf{C}_3}{nw^2}\min_{\alpha\in\left]0;1\right[}\left\{\frac{1}{\epsilon^2\left(1-\alpha\right)^4}+\frac{1}{\alpha^2}\right\}\\&\leq\frac{\mathsf{C}_3}{nw^2}\min_{\alpha\in\left]0;1\right[}\left\{\frac{1}{\epsilon^2\left(1-\alpha\right)^4}+\frac{1}{\alpha^4}\right\}\\&\leq\frac{\mathsf{C}_3}{nw^2}\left(1+\epsilon^{-2/5}\right)^5.\end{aligned}$$

Setting $\epsilon' = \frac{\epsilon}{w}$ and $\mathsf{C}_{3;1/w}(\epsilon') = \epsilon'^2 \times \frac{\mathsf{C}_3}{w^2}\left(1+(w\epsilon')^{-2/5}\right)^5 = \frac{C_3}{w^4}\left(1+(w\epsilon')^{2/5}\right)^5$, the result holds.

□

Lemma 3.4.4. (Product) *Consider u, v two positive scalars, and (U_n), (V_n) some random sequences. Assume that there exists nonnegative constants $\tilde{\mathsf{C}}_4$ and $\breve{\mathsf{C}}_4$ such that $\forall \epsilon > 0, \forall n \geq 1$:*

$$\mathbb{P}(|U_n - u| > \epsilon) \leq \frac{\tilde{\mathsf{C}}_4}{n}\frac{1}{\epsilon^2} \quad \textit{and} \quad \mathbb{P}(|V_n - v| > \epsilon) \leq \frac{\breve{\mathsf{C}}_4}{n}\frac{1}{\epsilon^2}.$$

Then there exists a function $\mathsf{C}_{4;uv}$ such that $\forall \epsilon > 0$,

$$\mathbb{P}(|U_n V_n - uv| > \epsilon) \leq \frac{\mathsf{C}_{4;uv}(\epsilon)}{n}\frac{1}{\epsilon^2},$$

where $\mathsf{C}_{4;uv}(\epsilon) = \tilde{\mathsf{C}}_4\left(\frac{2uv+\epsilon}{u}\right)^2 + \breve{\mathsf{C}}_4(2u)^2$ is a positive function of ϵ depending on u, v, $\tilde{\mathsf{C}}_4$ and $\breve{\mathsf{C}}_4$.

Proof of Lemma 3.4.4. By straightforward inequalities, we get

$$\begin{aligned}
\mathbb{P}(|U_n V_n - uv| > \epsilon) &= \mathbb{P}(|U_n V_n - uV_n + uV_n - uv| > \epsilon) \\
&\leq \mathbb{P}\left(V_n |U_n - u| > \frac{\epsilon}{2},\, u|V_n - v| \leq \frac{\epsilon}{2}\right) \\
&\quad + \mathbb{P}\left(u|V_n - v| > \frac{\epsilon}{2}\right) \\
&\leq \mathbb{P}\left(\left(v + \frac{\epsilon}{2u}\right)|U_n - u| > \frac{\epsilon}{2}\right) + \mathbb{P}\left(|V_n - v| > \frac{\epsilon}{2u}\right) \\
&\leq \mathbb{P}\left(|U_n - u| > \frac{\epsilon u}{2uv + \epsilon}\right) + \mathbb{P}\left(|V_n - v| > \frac{\epsilon}{2u}\right) \\
&\leq \frac{\tilde{\mathsf{C}}_4}{n}\left(\frac{2uv+\epsilon}{\epsilon u}\right)^2 + \frac{\breve{\mathsf{C}}_4}{n}\left(\frac{2u}{\epsilon}\right)^2 \\
&\leq \frac{\mathsf{C}_{4;uv}(\epsilon)}{n}\frac{1}{\epsilon^2}.
\end{aligned}$$

□

Lemma 3.4.5. Proximity between σ^2, α^2, β^2, δ^2 and their estimators
$\forall n \geq 1$ and $\forall \epsilon > 0$, we have :

- *for $\hat{\sigma}_n^2$ and σ^2:*

$$\mathbb{P}\left(\left|\hat{\sigma}_n^2 - \sigma^2\right| > \epsilon\right) \leq \frac{\sqrt{K_2}}{n}\frac{1}{\epsilon^2}.$$

- *for $\hat{\delta}_n^2$ and δ^2:*

$$\mathbb{P}\left(\left|\hat{\delta}_n^2 - \delta^2\right| > \epsilon\right) \leq \frac{C_{\delta^2}}{n\epsilon^2},$$

with

$$\begin{aligned}
C_{\delta^2} &= 2K_4 + (100 + K_1^2)K_2 + 2^4\sqrt{6}K_2^{5/4} + 4K_2^{3/2} + 2^2 3K_2^2 \\
&\quad + 4K_2^{1/2}\left(K_2^{1/4} + 2\sqrt{6}\right)\sqrt{K_1^2 K_2 + 4K_2(1+3K_2) + 2K_4}
\end{aligned}$$

- *for $\hat{\beta}_n^2$ and β^2:*

$$\mathbb{P}\left(\left|\hat{\beta}_n^2 - \beta^2\right| > \epsilon\right) \leq \frac{C_{\beta^2}(\epsilon)}{n\epsilon^2},$$

with $C_{\beta^2}(\epsilon) = 4K_1^2\sqrt{K_2} + \mathsf{C}_{\delta^2} + 2K_1\sqrt{K_2}\,\epsilon$.

- *for $\hat{\alpha}_n^2$ and α^2:*

$$\mathbb{P}\left(\left|\hat{\alpha}_n^2 - \alpha^2\right| > \epsilon\right) \leq \frac{C_{\alpha^2}(\epsilon)}{n\epsilon^2},$$

with $C_{\alpha^2}(\epsilon) = 2^3 C_{\delta^2} + 2^4 K_1^2 \sqrt{K_2} + 2^2 K_1 \sqrt{K_2}\,\epsilon$.

Proof of Lemma 3.4.5.
Consider $\hat{\sigma}_n^2$ and σ^2.
Recall that $\hat{\sigma}_n^2 = \frac{1}{q}\sum_{j=1}^q \left(\frac{1}{n}\sum_{i=1}^n y_{ij}^2\right)$ and $\sigma^2 = \frac{1}{q}\sum_{j=1}^q \mathbb{E}\left[y_{1j}^2\right] = \frac{1}{q}\sum_{j=1}^q \mu_j$.
Following the ideas of Ledoit and Wolf [115] who obtain the convergence of the fourth order moment, we rather control the second order moment as follows :

$$\begin{aligned}
\mathbb{E}\left[\left(\hat{\sigma}_n^2 - \sigma^2\right)^2\right] &= \mathbb{E}\left[\left(\frac{1}{q}\sum_{j=1}^q \frac{1}{n}\sum_{i=1}^n \left(y_{ij}^2 - \mu_j\right)\right)^2\right] \\
&= \mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^n \frac{1}{q}\sum_{j=1}^q \left(y_{ij}^2 - \mu_j\right)\right)^2\right] \\
&= \frac{1}{n^2}\sum_{i_1=1}^n\sum_{i_2=1}^n \mathbb{E}\left[\frac{1}{q}\sum_{j=1}^q \left(y_{i_1 j}^2 - \mu_j\right) \times \frac{1}{q}\sum_{j=1}^q \left(y_{i_2 j}^2 - \mu_j\right)\right].
\end{aligned}$$

This last expression is equal to zero for any $i_1 \neq i_2$ because of the independence between observations. Thus we get

$$\begin{aligned}
\mathbb{E}\left[\left(\hat{\sigma}_n^2 - \sigma^2\right)^2\right] &= \frac{1}{n^2}\sum_{i=1}^n \mathbb{E}\left[\left(\frac{1}{q}\sum_{j=1}^q \left(y_{1j}^2 - \mu_j\right)\right)^2\right] \\
&= \frac{1}{n}\mathbb{E}\left[\left(\frac{1}{q}\sum_{j=1}^q \left(y_{1j}^2 - \mu_j\right)\right)^2\right] \\
&= \frac{1}{n}\left(\mathbb{E}\left[\left(\frac{1}{q}\sum_{j=1}^q y_{1j}^2\right)^2\right] - \left(\mathbb{E}\left[\frac{1}{q}\sum_{j=1}^q y_{1j}^2\right]\right)^2\right) \\
&\leq \frac{1}{n}\mathbb{E}\left[\left(\frac{1}{q}\sum_{j=1}^q y_{1j}^2\right)^2\right] \leq \frac{1}{n}\left(\mathbb{E}\left[\left(\frac{1}{q}\sum_{j=1}^q y_{1j}^2\right)^4\right]\right)^{1/2} \\
&\leq \frac{1}{n}\left(\frac{1}{q}\sum_{j=1}^q \mathbb{E}\left[y_{1j}^8\right]\right)^{1/2}.
\end{aligned}$$

Therefore, using the second assumption (A_2), one gets

$$\mathbb{E}\left[\left(\hat{\sigma}_n^2 - \sigma^2\right)^2\right] \leq \frac{\sqrt{K_2}}{n}. \tag{3.16}$$

Finally, we have by Markov inequality the bound

$$\mathbb{P}\left[\left|\hat{\sigma}_n^2 - \sigma^2\right| > \epsilon\right] \leq \frac{\mathbb{E}\left[\left(\hat{\sigma}_n^2 - \sigma^2\right)^2\right]}{\epsilon^2} \leq \frac{\sqrt{K_2}}{n\epsilon^2}.$$

Consider $\hat{\delta}_n^2$ and δ^2.
Combining the expressions (A.2) and (A.3) on page 394 in Ledoit and Wolf (2004) ([115]) we get

$$\hat{\delta}_n^2 - \delta^2 = \left(\hat{\sigma}_n^2 - \sigma^2\right)^2 - 2\sigma^2\left(\hat{\sigma}_n^2 - \sigma^2\right) + \left\|S_n^2\right\|^2 - \mathbb{E}\left(\left\|S_n^2\right\|^2\right). \tag{3.17}$$

Similarly using their expressions, from page 394 (A.4) to page 399, and page 390 (A.1), we have respectively the inequalities

$$\begin{aligned} &\mathrm{Var}\left(\left\|S_n^2\right\|^2\right) \leq \frac{1}{n}\left(K_1^2 K_2 + 4K_2\left(1 + 3K_2\right) + 2K_4\right) \\ &\sigma^2 \leq \sqrt{K_2}. \end{aligned} \tag{3.18}$$

Combining these expressions with Bienaymé-Tchebychev, Markov and Cauchy-Schwartz inequalities, we obtain a control of $\mathbb{P}\left(\left|\hat{\delta}_n^2 - \delta^2\right| > \epsilon\right)$ by a function of n, ϵ, A_2, A_4 and $\mathrm{Var}(\|S_n^2\|^2)$ where $A_k = \mathbb{E}\left(\left|\hat{\sigma}_n^2 - \sigma^2\right|^k\right)$. Indeed we have, for all $\epsilon > 0$,

$$\begin{aligned} \mathbb{P}\left(\left|\hat{\delta}_n^2 - \delta^2\right| > \epsilon\right) &\leq \frac{1}{\epsilon^2}\mathbb{E}\left(\left(\hat{\delta}_n^2 - \delta^2\right)^2\right) \\ &\leq \frac{1}{\epsilon^2}\left\{\mathbb{E}\left[\left(\hat{\sigma}_n^2 - \sigma^2\right)^4\right] + 4\sigma^4\mathbb{E}\left[\left(\hat{\sigma}_n^2 - \sigma^2\right)^2\right]\right. \\ &\quad + \mathbb{E}\left[\left(\left\|S_n^2\right\|^2 - \mathbb{E}\left\|S_n^2\right\|^2\right)^2\right] \\ &\quad + 4\sigma^2\mathbb{E}\left[\left|\hat{\sigma}_n^2 - \sigma^2\right|^3\right] + 4\sigma^2\mathbb{E}\left[\left|\hat{\sigma}_n^2 - \sigma^2\right|\left(\left\|S_n^2\right\|^2 - \mathbb{E}\left(\left\|S_n^2\right\|^2\right)\right)\right] \\ &\quad \left. + 2\mathbb{E}\left[\left(\hat{\sigma}_n^2 - \sigma^2\right)^2\left|\left\|S_n^2\right\|^2 - \mathbb{E}\left(\left\|S_n^2\right\|^2\right)\right|\right]\right\} \\ &\leq \frac{1}{\epsilon^2}\left\{A_4 + 4\sigma^4 A_2 + \mathrm{Var}\left(\left\|S_n^2\right\|^2\right) + 4\sigma^2\sqrt{A_2 A_4}\right. \\ &\quad \left. + 4\sigma^2\sqrt{A_2\mathrm{Var}\left(\left\|S_n^2\right\|^2\right)} + 2\sqrt{A_4\mathrm{Var}\left(\left\|S_n^2\right\|^2\right)}\right\}. \end{aligned}$$

Now by some previous controls established by Ledoit and Wolf (2004) ([115], page 394) we have

$$A_4 \leq \frac{96K_2}{n},$$

$$\mathrm{Var}\left(\left\|S_n^2\right\|^2\right) \leq \frac{1}{n}\left(K_1^2 K_2 + 4K_2\left(1 + 3K_2\right) + 2K_4\right) = \frac{1}{n}K$$

and

$$\sigma^2 \leq \sqrt{K_2}.$$

Using the control stated in (3.16), $A_2 \leq \sqrt{K_2}/n$, we can easily get the explicit constant C_{δ^2} as a function of K_1, K_2, and K_4.

For all $\epsilon > 0$, for all $n \in \mathbb{N}^*$, we have

$$\begin{aligned}
\mathbb{P}\left(\left|\hat{\delta}_n^2 - \delta^2\right| > \epsilon\right) &\leq \frac{1}{n\epsilon^2}\left[96K_2 + 4K_2K_2^{1/2} + K + 4K_2^{1/2}\sqrt{96K_2^{1/2}K_2}\right. \\
&\quad \left. + 4K_2^{1/2}\sqrt{K_2^{1/2}K} + 2\sqrt{96K_2K}\right] \\
&\leq \frac{1}{n\epsilon^2}\left\{2K_4 + (100 + K_1^2)K_2 + 2^4\sqrt{6}K_2^{5/4} + 4K_2^{3/2} + 2^2 3K_2^2\right. \\
&\quad \left. + 4K_2^{1/2}\left(K_2^{1/4} + 2\sqrt{6}\right)\sqrt{K_1^2K_2 + 4K_2\left(1 + 3K_2\right) + 2K_4}\right\} \\
&\leq \frac{\mathsf{C}_{\delta^2}}{n\epsilon^2}.
\end{aligned}$$

Consider $\hat{\beta}_n^2$ and β^2.
Since $\delta^2 = \alpha^2 + \beta^2$ yielding $\delta^2 \geq \beta^2$, Ledoit and Wolf (2004) showed ([115], proof of Lemma 3.4 page 401, lines from -12 to -6) that

$$-\max\left(|\bar{\beta}_n^2 - \beta^2|, |\hat{\delta}_n^2 - \delta^2|\right) \leq \hat{\beta}_n^2 - \beta^2 \leq |\bar{\beta}_n^2 - \beta^2|.$$

From this we deduce

$$\begin{aligned}
|\hat{\beta}_n^2 - \beta^2| &\leq \max\left\{\max\left(|\bar{\beta}_n^2 - \beta^2|, |\hat{\delta}_n^2 - \delta^2|\right), |\bar{\beta}_n^2 - \beta^2|\right\} \\
&\leq \max\left(|\bar{\beta}_n^2 - \beta^2|, |\hat{\delta}_n^2 - \delta^2|\right).
\end{aligned}$$

Controlling $|\bar{\beta}_n^2 - \beta^2|$ leads to a control for $|\hat{\delta}_n^2 - \delta^2|$ and $|\hat{\beta}_n^2 - \beta^2|$. By the same arguments as in Ledoit and Wolf (2004) [115] (proof of Lemma 3.4, page 399, equation (A.7)), we have the following expression

$$\bar{\beta}_n^2 - \beta^2 = \frac{1}{n}\|S_n^2 - S^2\|^2 + \left(\frac{1}{n^2}\sum_{i=1}^n \|Z_iZ_i' - S^2\|^2 - \mathbb{E}\left[\frac{1}{n^2}\sum_{i=1}^n \|Z_iZ_i' - S^2\|^2\right]\right).$$

Now, splitting the probability into two terms, on the one hand, using Markov inequality on the first term and applying Bienaymé-Tchebychev inequality to the second term, we get

$$\mathbb{P}\left(|\bar{\beta}_n^2 - \beta^2| > \epsilon\right) \leq \frac{2}{\epsilon}\mathbb{E}\left(\frac{1}{n}\|S_n^2 - S^2\|^2\right) + \frac{4}{\epsilon^2}\mathrm{Var}\left(\frac{1}{n^2}\sum_{i=1}^n \|Z_iZ_i' - S^2\|^2\right).$$

Following Ledoit and Wolf ([115], proof of Lemma 3.1 page 391 line +5), we have

$$\mathbb{E}\left(\left\|S_n^2 - S^2\right\|^2\right) \leq K_1\sqrt{K_2}.$$

Moreover, we have (in the proof of Lemma 3.4, page 401 line +3)

$$\mathrm{Var}\left(\frac{1}{n^2}\sum_{i=1}^n \left\|Z_iZ_i' - S^2\right\|^2\right) \leq K_1^2\sqrt{K_2}/n.$$

We obtain

$$\mathbb{P}\left(\left|\bar{\beta}_n^2 - \beta^2\right| > \epsilon\right) \leq \frac{2}{\epsilon}\frac{K_1\sqrt{K_2}}{n} + \frac{4}{\epsilon^2}\frac{K_1^2\sqrt{K_2}}{n}.$$

Finally, with $\mathbb{P}\left(\left|\hat{\delta}_n^2 - \delta^2\right| > \epsilon\right) \leq \frac{C_{\delta^2}}{n\epsilon^2}$ and

$$\mathbb{P}\left(\left|\hat{\beta}_n^2 - \beta^2\right| > \epsilon\right) \leq \mathbb{P}\left(\left|\bar{\beta}_n^2 - \beta^2\right| > \epsilon\right) + \mathbb{P}\left(\left|\hat{\delta}_n^2 - \delta^2\right| > \epsilon\right),$$

we obtain

$$\mathbb{P}\left(\left|\hat{\beta}_n^2-\beta^2\right|>\epsilon\right)\leq\frac{1}{n\epsilon^2}\left(4K_1^2\sqrt{K_2}+\mathsf{C}_{\delta^2}+2K_1\sqrt{K_2}\,\epsilon\right)\leq\frac{C_{\beta^2}(\epsilon)}{n\epsilon^2}.$$

Remark that $C_{\beta^2}(\epsilon)$ tends to $4K_1^2\sqrt{K_2}+\mathsf{C}_{\delta^2}$ when ϵ tends to 0.

Consider $\hat{\alpha}_n^2$ and α^2.
Since we have $\hat{\alpha}_n^2=\hat{\delta}_n^2-\hat{\beta}_n^2$ and $\alpha^2+\beta^2=\delta^2$, one can easily see that $\hat{\alpha}_n^2-\alpha^2=\hat{\delta}_n^2-\hat{\beta}_n^2-\delta^2+\beta^2$. For all $\epsilon>0$, we get

$$\begin{aligned}\mathbb{P}\left(\left|\hat{\alpha}_n^2-\alpha^2\right|>\epsilon\right) &\leq \mathbb{P}\left(\left|\hat{\delta}_n^2-\delta^2\right|>\frac{\epsilon}{2}\right)+\mathbb{P}\left(\left|\hat{\beta}_n^2-\beta^2\right|>\frac{\epsilon}{2}\right)\\ &\leq \frac{2^2\mathsf{C}_{\delta^2}}{n\epsilon^2}+\frac{2^2\mathsf{C}_{\beta^2}(\epsilon/2)}{n\epsilon^2}\\ &\leq \frac{1}{n\epsilon^2}\left(2^3\mathsf{C}_{\delta^2}+2^4K_1^2\sqrt{K_2}+2^2K_1\sqrt{K_2}\,\epsilon\right)\\ &\leq \frac{C_{\alpha^2}(\epsilon)}{n\epsilon^2}.\end{aligned}$$

Remark that $C_{\alpha^2}(\epsilon)$ tends to $2^3\mathsf{C}_{\delta^2}+2^4K_1^2\sqrt{K_2}$ when ϵ tends to 0.

□

In the next lemma 3.4.6, we control the proximity between $1/\hat{\rho}_n^*$ and $1/\rho^*$, that we denote $g_n(\epsilon)$ and show that it is of order $O(1/n)$. For this, we first apply product lemma 3.4.4 to $\hat{\beta}_n^2$ and $\hat{\sigma}_n^2$. Then, we apply the inverse lemma 3.4.3 to $\hat{\beta}_n^2\hat{\sigma}_n^2$. Finally, we use another time product lemma 3.4.4 applied to $\hat{\alpha}_n^2$ and $1/\hat{\beta}_n^2\hat{\sigma}_n^2$.

Lemma 3.4.6. Proximity between $1/\rho^*$ and $1/\hat{\rho}_n^*$
For any $\epsilon>0$, we have

$$g_n(\epsilon)=\mathbb{P}\left(\left|\frac{1}{\hat{\rho}_n^*}-\frac{1}{\rho^*}\right|>\epsilon\right)\leq\frac{\mathsf{G}(\epsilon)}{n\epsilon^2}$$

with

$$\mathsf{G}(\epsilon)=\mathsf{C}_{3;1/\beta^2\sigma^2}(\epsilon)\left(2\alpha^2+\epsilon\beta^2\sigma^2\right)^2+\frac{2^2\mathsf{C}_{\alpha^2}(\epsilon)}{\beta^4\sigma^4}$$

and

$$\mathsf{C}_{3;1/\beta^2\sigma^2}(\epsilon)=\left[K_2^{1/2}\frac{\left(2\sigma^2\beta^2+\epsilon\right)^2}{\beta^8\sigma^{12}}+\frac{2^2\mathsf{C}_{\beta^2}(\epsilon)}{\beta^8\sigma^4}\right]\left(1+\left(\beta^2\sigma^2\epsilon\right)^{2/5}\right)^5,$$

with C_{β^2} and C_{α^2} defined in lemma 3.4.5.

Remark : the function $\mathsf{C}_{3;1/\beta^2\sigma^2}(\epsilon)$ may be clearly bounded by a polynomial of degree 4 in ϵ. As a consequence, the function $G(\epsilon)$ may be bounded by a polynomial of degree 6.

Proof of Lemma 3.4.6. We apply the product lemma 3.4.4 to obtain a control for $\hat{\beta}_n^2\hat{\sigma}_n^2$ thanks to lemma 3.4.5 which gives us some control of $\hat{\sigma}_n^2$ and $\hat{\beta}_n^2$. For all $\epsilon>0$, one gets

$$\mathbb{P}\left(\left|\hat{\beta}_n^2\hat{\sigma}_n^2-\beta^2\sigma^2\right|>\epsilon\right)\leq\frac{C_{4;\beta^2\sigma^2}(\epsilon)}{n\epsilon^2},\tag{3.19}$$

with

$$C_{4;\sigma^2\beta^2}(\epsilon)=K_2^{1/2}\left(\frac{2\sigma^2\beta^2+\epsilon}{\sigma^2}\right)^2+C_{\beta^2}(\epsilon)\left(2\sigma^2\right)^2.\tag{3.20}$$

We now apply the inverse lemma 3.4.3 with inequality (3.19) and obtain a control of $1/\hat{\beta}_n^2\hat{\sigma}_n^2$. That is, for all $\epsilon > 0$, we have

$$\mathbb{P}\left(\left|\frac{1}{\hat{\beta}_n^2\hat{\sigma}_n^2} - \frac{1}{\beta^2\sigma^2}\right| > \epsilon\right) \leq \frac{C_{3;1/\beta^2\sigma^2}(\epsilon)}{n\epsilon^2},$$

with $\mathsf{C}_{3;1/\beta^2\sigma^2}$ defined by

$$\mathsf{C}_{3;1/\beta^2\sigma^2}(\epsilon) = \frac{\mathsf{C}_{4;\beta^2\sigma^2}(\epsilon)}{\beta^8\sigma^8}\left(1 + \left(\beta^2\sigma^2\epsilon\right)^{2/5}\right)^5.$$

Applying the product lemma 3.4.4 with $u = 1/(\beta^2\sigma^2)$ and $v = \alpha^2$, we obtain

$$\mathsf{C}_{4;1/\rho^*}(\epsilon) = \mathsf{C}_{3;1/\beta^2\sigma^2}(\epsilon)\left(2\alpha^2 + \epsilon\beta^2\sigma^2\right)^2 + \mathsf{C}_{\alpha^2}(\epsilon)\frac{2^2}{\beta^4\sigma^4}.$$

Remark that $\mathsf{C}_{4;1/\rho^*}$ tends to

$$\frac{2^4\alpha^4K_2^{1/2}}{\beta^4\sigma^8} + \frac{2^6\alpha^4\left(2^2K_1^2\sqrt{K_2} + \mathsf{C}_{\delta^2}\right)}{\beta^8\sigma^4} + \frac{2^5\left(2K_1^2\sqrt{K_2} + \mathsf{C}_{\delta^2}\right)}{\beta^4\sigma^4}$$

when ϵ tends to 0. □

Proof of theorem 3.3.1.
Recall that $\hat{a}_n^* = 1 + \frac{K_3}{\hat{\rho}_n^*}$ and $a^* = 1 + \frac{K_3}{\rho^*}$. For any $u > 2n$, we have

$$\begin{aligned}\mathbb{P}\left(n\bar{Z}_n'\hat{\Sigma}_n^{*-2}\bar{Z}_n \geq u\left(1 + \hat{a}_n^* + 2\epsilon\right)\right) &\leq \mathbb{P}\left(n\bar{Z}_n'\hat{\Sigma}_n^{*-2}\bar{Z}_n \geq u\left(1 + a^* + \epsilon\right)\right)\\ &\quad + \mathbb{P}\left(|\hat{a}_n - a^*| \geq \epsilon\right)\\ &\leq \text{(I)} + \text{(II)}. \end{aligned} \tag{3.21}$$

We start by establishing a control for (I). Define $\Delta_n = n\bar{Z}_n'\left(\hat{\Sigma}_n^{*-2} - \Sigma_n^{*-2}\right)\bar{Z}_n$, then we have

$$\text{(I)} = \mathbb{P}\left(n\bar{Z}_n'\Sigma_n^{*-2}\bar{Z}_n + \Delta_n \geq u\left(1 + a^* + \epsilon\right)\right).$$

Since $u > 2n > n$, we have

$$\begin{aligned}\text{(I)} &\leq \mathbb{P}\left(n\bar{Z}_n'\Sigma_n^{*-2}\bar{Z}_n + \Delta_n \geq u\left(1 + a^* + \epsilon\right), |\Delta_n| \leq \epsilon n\right) + \mathbb{P}\left(|\Delta_n| > \epsilon n\right)\\ &\leq \mathbb{P}\left(n\bar{Z}_n'\Sigma_n^{*-2}\bar{Z}_n \geq u\left(1 + a^* + \epsilon\right) - \epsilon n\right) + \mathbb{P}\left(|\Delta_n| > \epsilon n\right)\\ &\leq \mathbb{P}\left(n\bar{Z}_n'\Sigma_n^{*-2}\bar{Z}_n \geq u\left(1 + a^*\right)\right) + \mathbb{P}\left(|\Delta_n| > \epsilon n\right). \end{aligned} \tag{3.22}$$

Theorem 3.2.3 gives us an exponential bound controlling the first term of the right hand of the inequality when $a = a^*$ and $u > 2n$.
Now use the following matrix factorisation $A^{-1} - B^{-1} = A^{-1}(B - A)B^{-1}$ to control the second term in the right hand with $A = \hat{\Sigma}_n^{*2}$ and $B = \Sigma_n^{*2}$. It is easy to see that $B - A = (\rho^* - \hat{\rho}_n^*)I_q$, then we obtain

$$\begin{aligned}\Delta_n &= Tr(\Delta_n)\\ &= Tr\left(n\bar{Z}_n'\left(\hat{\Sigma}_n^{*-2} - \Sigma_n^{*-2}\right)\bar{Z}_n\right)\\ &= n\left(\rho^* - \hat{\rho}_n^*\right)Tr\left(\bar{Z}_n'\hat{\Sigma}_n^{*-2}\Sigma_n^{*-2}\bar{Z}_n\right).\end{aligned}$$

Recall that

$$\Sigma_n^{*2} = S_n^2 + \rho^*I_q = O_n'\Lambda_n^2O_n + \rho^*I_q = O_n'\left(\Lambda_n^2 + \rho^*I_q\right)O_n.$$

then

$$\Sigma_n^{*-2} = O_n' \begin{pmatrix} \frac{1}{\lambda_1+\rho^*} & & & & 0 \\ & \ddots & & & \\ & & \frac{1}{\lambda_n+\rho^*} & & \\ & & & \frac{1}{\rho^*} & \\ & 0 & & & \ddots \\ & & & & & \frac{1}{\rho^*} \end{pmatrix} O_n.$$

Using the same rotation matrix O_n, we obtain $\hat{\Sigma}_n^{*-2}\Sigma_n^{*-2} = O_n' D O_n$, with

$$D = \begin{pmatrix} \frac{1}{(\lambda_1+\rho^*)(\lambda_1+\hat{\rho}_n^*)} & & & & 0 \\ & \ddots & & & \\ & & \frac{1}{(\lambda_n+\rho^*)(\lambda_n+\hat{\rho}_n^*)} & & \\ & & & \frac{1}{\rho^*\hat{\rho}_n^*} & \\ & 0 & & & \ddots \\ & & & & & \frac{1}{\rho^*\hat{\rho}_n^*} \end{pmatrix}.$$

It follows that

$$\begin{aligned} \Delta_n &= n\,(\rho^* - \hat{\rho}_n^*)\, Tr\left(\bar{Z}_n' O_n' D^{\frac{1}{2}} D^{\frac{1}{2}} O_n \bar{Z}_n\right) \\ &= (\rho^* - \hat{\rho}_n^*)\, Tr\left(\left(D^{\frac{1}{2}} n^{\frac{1}{2}} O_n \bar{Z}_n\right)' \left(D^{\frac{1}{2}} n^{\frac{1}{2}} O_n \bar{Z}_n\right)\right) \\ &= (\rho^* - \hat{\rho}_n^*) \left\| D^{\frac{1}{2}} n^{\frac{1}{2}} \bar{Y}_n \right\|_2^2. \end{aligned}$$

Since, for any x in $\mathbb{R}^q$, $\|D^{\frac{1}{2}} x\|_2^2 \leq \frac{1}{\rho^*\hat{\rho}_n^*}\|x\|_2^2$, and because we have $\|x\|_2^2 = q\|x\|^2$, we get

$$\begin{aligned} |\Delta_n| &\leq \frac{|\rho^* - \hat{\rho}_n^*|}{\rho^*\hat{\rho}_n^*} \left\| n^{\frac{1}{2}} \bar{Y}_n \right\|_2^2 \\ &\leq \left| \frac{1}{\rho^*} - \frac{1}{\hat{\rho}_n^*} \right| q \left\| n^{\frac{1}{2}} \bar{Y}_n \right\|^2. \end{aligned}$$

Lemma 3.4.6 gives a control of the first term on the right-hand side of this inequality so that it is sufficient to control the second term. Write

$$\begin{aligned} \left\| n^{\frac{1}{2}} \bar{Y}_n \right\|^2 &= \frac{1}{qn} \sum_{j=1}^{q} \left(\sum_{i=1}^{n} Y_{i,j} \right)^2 \\ &= \frac{1}{qn} \sum_{j=1}^{q} \sum_{i=1}^{n} Y_{i,j}^2 + \frac{1}{qn} \sum_{j=1}^{q} \sum_{i=1}^{n} \sum_{\substack{i'=1 \\ i' \neq i}}^{n} Y_{i,j} Y_{i',j} \\ &= \mathrm{I}_1 + \mathrm{I}_2 \end{aligned} \tag{3.23}$$

Since $\mathbb{E}(\mathrm{I}_1) = \mathbb{E}\left(\frac{1}{qn}\sum_{j=1}^{q}\sum_{i=1}^{n} Y_{i,j}^2\right) = \sigma^2$, use Bienaymé-Tchebychev inequality and the independence

of the Y_i's to get

$$\begin{aligned}
\mathbb{P}\left(\frac{1}{qn}\sum_{j=1}^{q}\sum_{i=1}^{n}Y_{i,j}^2-\sigma^2>\frac{\epsilon}{2}\right) &\leq \mathbb{P}\left(\left|\frac{1}{qn}\sum_{j=1}^{q}\sum_{i=1}^{n}Y_{i,j}^2-\sigma^2\right|>\frac{\epsilon}{2}\right)\\
&\leq \frac{4}{\epsilon^2}\mathrm{Var}\left(\frac{1}{qn}\sum_{j=1}^{q}\sum_{i=1}^{n}Y_{i,j}^2\right)\\
&\leq \frac{4}{\epsilon^2}\frac{1}{nq^2}\mathbb{E}\left(\left(\sum_{j=1}^{q}Y_{1,j}^2\right)^2\right). \qquad (3.24)
\end{aligned}$$

Then, by hypothesis (A_2), we have $\mathbb{E}\left(\frac{1}{q}\sum_{j=1}^{q}Y_{1,j}^4\right)\leq\sqrt{K_2}$. Then, by Cauchy-Schwartz inequality, we obtain

$$\begin{aligned}
\frac{1}{nq^2}\mathbb{E}\left(\left(\sum_{j=1}^{q}Y_{1,j}^2\right)^2\right) &\leq \frac{1}{nq}\mathbb{E}\left(\frac{1}{q}\sum_{j=1}^{q}Y_{1,j}^4\right)+\frac{1}{nq^2}\sum_{j=1}^{q}\sum_{\substack{k=1\\k\neq j}}^{q}\mathbb{E}\left(Y_{1,j}^2Y_{1,k}^2\right)\\
&\leq \frac{1}{nq}\sqrt{K_2}+\frac{1}{nq^2}\sum_{j=1}^{q}\sum_{\substack{k=1\\k\neq j}}^{q}\sqrt{\mathbb{E}\left(Y_{1,j}^4\right)}\sqrt{\mathbb{E}\left(Y_{1,k}^4\right)}\\
&\leq \frac{1}{nq}\sqrt{K_2}+\frac{1}{n}\left(\frac{1}{q}\sum_{j=1}^{q}\sqrt{\mathbb{E}\left(Y_{1,j}^4\right)}\right)^2\\
&\leq \frac{1}{nq}\sqrt{K_2}+\frac{1}{n}\mathbb{E}\left(\frac{1}{q}\sum_{j=1}^{q}Y_{1,j}^4\right)\\
&\leq \frac{1}{n}\sqrt{K_2}\left(\frac{1}{q}+1\right). \qquad (3.25)
\end{aligned}$$

Finally, combining inequalities (3.24, 3.25), we get the following control for I_1

$$\mathbb{P}\Big(\ \mathrm{I}_1-\mathbb{E}(\mathrm{I}_1)>\frac{\eta}{2}\Big) \leq \frac{4}{\eta^2}\frac{1}{n}\sqrt{K_2}\left(\frac{1}{q}+1\right). \qquad (3.26)$$

Now, we focus on I_2. Using the independence between the observations Y_i's, we have

$$\mathbb{E}\left(\mathrm{I}_2\right)=\mathbb{E}\left(\frac{1}{qn}\sum_{j=1}^{q}\sum_{i=1}^{n}\sum_{\substack{i'=1\\i'\neq i}}^{n}Y_{i,j}Y_{i',j}\right)=0.$$

By Bienaymé-Tchebychev inequality, we have

$$\mathbb{P}\left(\mathrm{I}_2>\frac{\eta}{2}\right) \leq \frac{4}{\eta^2}\mathbb{E}\left[\left(\frac{1}{q}\sum_{j=1}^{q}\frac{1}{n}\sum_{i=1}^{n}\sum_{\substack{i'=1\\i'\neq i}}^{n}Y_{i,j}Y_{i',j}\right)^2\right]. \qquad (3.27)$$

Furthermore, since $\frac{1}{n^2} = \frac{(n-1)^2}{4}\left(\frac{2}{n(n-1)}\right)^2$, we can express the expectation above as the expectation of a U-statistic

$$\mathbb{E}\left[\left(\frac{1}{q}\sum_{j=1}^{q}\frac{1}{n}\sum_{i=1}^{n}\sum_{\substack{i'=1\\ i'\neq i}}^{n}Y_{i,j}Y_{i',j}\right)^2\right] = \frac{(n-1)^2}{4}\mathbb{E}\left[\left(\frac{2}{n(n-1)}\sum_{i=1}^{n}\sum_{\substack{i'=1\\ i'\neq i}}^{n}\frac{1}{q}\sum_{j=1}^{q}Y_{i,j}Y_{i',j}\right)^2\right].$$

More precisely, this is a U-statistic of degree 2 with kernel $w\left(Y_i, Y_i'\right) = \frac{1}{q}\sum_{j=1}^{q}Y_{i,j}Y_{i',j}$, with $\mathbb{E}\left[w\left(Y_i, Y_i'\right)\right] = 0$ and degenerated gradients

$$\mathbb{E}\left[w\left(Y_i, Y_{i'}\right) \mid Y_i\right] = 0 \quad \text{and} \quad \mathbb{E}\left[w\left(Y_i, Y_{i'}\right) \mid Y_{i'}\right] = 0,$$

where $\mathbb{E}(Z|Y)$ denotes the expectation of Z conditionally to Y. Using the expression of the variance of this U-statistic as given in Lee (2019) [117], it follows that

$$\begin{aligned}\mathbb{E}\left[\left(\frac{1}{q}\sum_{j=1}^{q}\frac{1}{n}\sum_{i=1}^{n}\sum_{\substack{i'=1\\ i'\neq i}}^{n}Y_{i,j}Y_{i',j}\right)^2\right] &= \frac{(n-1)^2}{4}\frac{1}{\frac{n(n-1)}{2}}\binom{n-2}{0}\mathrm{Var}\left(w\left(Y_i, Y_{i'}\right)\right)\\ &= \frac{n-1}{2n}\mathbb{E}\left[\left(\frac{1}{q}\sum_{j=1}^{q}Y_{1,j}Y_{2,j}\right)^2\right]. \end{aligned} \tag{3.28}$$

Now, we have by independence

$$\begin{aligned}\mathbb{E}\left[\left(\frac{1}{q}\sum_{j=1}^{q}Y_{1,j}Y_{2,j}\right)^2\right] &= \mathbb{E}\left[\frac{1}{q^2}\sum_{j=1}^{q}\sum_{k=1}^{q}Y_{1,j}Y_{2,j}Y_{1,k}Y_{2,k}\right]\\ &= \frac{1}{q^2}\sum_{j=1}^{q}\sum_{k=1}^{q}\left[\mathbb{E}\left(Y_{1,j}Y_{1,k}\right)\right]^2.\end{aligned}$$

Recall that $\mathbb{E}\left[Y_{1,j}Y_{1,k}\right] = 0$ if $j \neq k$. By using Hölder inequalities repetitively and by hypothesis (A_2), we have $\frac{1}{q}\sum_{j=1}^{q}\left[\mathbb{E}\left(Y_{1,j}^2\right)\right]^2 \leq \left(\frac{1}{q}\sum_{j=1}^{q}\mathbb{E}\left(Y_{1,j}^8\right)\right)^{\frac{1}{2}} \leq K_2^{\frac{1}{2}}$, yielding

$$\mathbb{E}\left[\left(\frac{1}{q}\sum_{j=1}^{q}Y_{1,j}Y_{2,j}\right)^2\right] \leq \frac{1}{q}\sqrt{K_2}. \tag{3.29}$$

Finally, combining equations (3.27,3.28) and (3.29), we obtain a control for I_2 as follows

$$\begin{aligned}\mathbb{P}\left(\mathrm{I}_2 > \frac{\eta}{2}\right) &= \mathbb{P}\left(\frac{1}{qn}\sum_{j=1}^{q}\sum_{i=1}^{n}\sum_{\substack{i'=1\\ i'\neq i}}^{n}Y_{i,j}Y_{i',j} > \frac{\eta}{2}\right)\\ &\leq \frac{1}{\eta^2}\frac{2(n-1)}{qn}\sqrt{K_2}.\end{aligned} \tag{3.30}$$

Finally, assumption (A_1) implies

$$\begin{aligned}
\mathbb{P}\left(|\Delta_n| > \epsilon n\right) &= \mathbb{P}\left(q\left|\frac{1}{\hat{\rho}_n^*} - \frac{1}{\rho^*}\right|\left\|n^{1/2}\bar{Y}_n\right\|^2 > \epsilon n\right) \\
&\leq \mathbb{P}\left(\left\|n^{\frac{1}{2}}\bar{Y}_n\right\|^2\left|\frac{1}{\hat{\rho}_n^*} - \frac{1}{\rho^*}\right| > \frac{\epsilon}{K_1}\right) \\
&\leq \mathbb{P}\left(\left(\left\|n^{\frac{1}{2}}\bar{Y}_n\right\|^2 - \sigma^2\right)\left|\frac{1}{\hat{\rho}_n^*} - \frac{1}{\rho^*}\right| > \frac{\epsilon}{2K_1}\right) + \mathbb{P}\left(\left|\frac{1}{\hat{\rho}_n^*} - \frac{1}{\rho^*}\right| > \frac{\epsilon}{2\sigma^2 K_1}\right).
\end{aligned}$$

Using the fact that $\mathbb{P}(AB > \epsilon) \leq \mathbb{P}(A > \sqrt{\epsilon}) + \mathbb{P}(B > \sqrt{\epsilon})$, and the definition of the function g_n in lemma 3.4.6, we have

$$\begin{aligned}
\mathbb{P}\left(|\Delta_n| > \epsilon n\right) &\leq \mathbb{P}\left(\left\|n^{\frac{1}{2}}\bar{Y}_n\right\|^2 - \sigma^2 > \sqrt{\frac{\epsilon}{2K_1}}\right) + g_n\left(\sqrt{\frac{\epsilon}{2K_1}}\right) + g_n\left(\frac{\epsilon}{2\sigma^2 K_1}\right) \\
&\leq \mathbb{P}\left(\mathrm{I}_1 - \sigma^2 > \frac{1}{2}\sqrt{\frac{\epsilon}{2K_1}}\right)) + \mathbb{P}\left(\mathrm{I}_2 > \frac{1}{2}\sqrt{\frac{\epsilon}{2K_1}}\right) \\
&\quad + g_n\left(\sqrt{\frac{\epsilon}{2K_1}}\right) + g_n\left(\frac{\epsilon}{2\sigma^2 K_1}\right).
\end{aligned}$$

Therefore, by inequalities (3.26) and (3.30), considering $\eta = \sqrt{\frac{\epsilon}{2K_1}}$, we get

$$\begin{aligned}
\mathbb{P}\left(|\Delta_n| > \epsilon n\right) &\leq \frac{4}{\left(\sqrt{\frac{\epsilon}{2K_1}}\right)^2} \times \left[\frac{\sqrt{K_2}}{n}\left(\frac{1}{q} + 1\right) + \frac{1}{2}\frac{n-1}{n}\frac{\sqrt{K_2}}{q}\right] \\
&\quad + g_n\left(\sqrt{\frac{\epsilon}{2K_1}}\right) + g_n\left(\frac{\epsilon}{2\sigma^2 K_1}\right) \\
&\leq \frac{4K_1\sqrt{K_2}}{\epsilon n}\left(2 + \frac{1}{q} + K_1\right) \\
&\quad + g_n\left(\sqrt{\frac{\epsilon}{2K_1}}\right) + g_n\left(\frac{\epsilon}{2\sigma^2 K_1}\right). \qquad (3.31)
\end{aligned}$$

We now complete the proof of the theorem by handling the term (II). By lemma 3.4.6, we get

$$\mathbb{P}\left(|\hat{a}_n - a^*| > \epsilon\right) = \mathbb{P}\left(\left|\frac{1}{\hat{\rho}_n^*} - \frac{1}{\rho^*}\right| > \frac{\epsilon}{K_3}\right) = g_n\left(\frac{\epsilon}{K_3}\right). \qquad (3.32)$$

With inequalities (3.21), (3.22), (3.31), and (3.32), and using the expression of G to bound g_n given in lemma 3.4.6, we finally obtain

$$\begin{aligned}
&\mathbb{P}\left(n\bar{Z}_n'\hat{\Sigma}_n^{*-2}\bar{Z}_n \geq u\left(1 + \hat{a}_n^* + 2\epsilon\right)\right) \leq \mathbb{P}\left(n\bar{Z}_n'\Sigma_n^{*-2}\bar{Z}_n \geq u\left(1 + a^*\right)\right) \\
&\quad + \frac{4K_1\sqrt{K_2}}{\epsilon n}\left(2 + \frac{1}{q} + K_1\right) + g_n\left(\sqrt{\frac{\epsilon}{2K_1}}\right) + g_n\left(\frac{\epsilon}{2\sigma^2 K_1}\right) + g_n\left(\frac{\epsilon}{K_3}\right) \\
&\leq \frac{2e^3}{9}\left(\frac{u-n}{2}\right)^{\frac{n}{2}}\frac{e^{-\frac{u-n}{2}}}{\Gamma\left(\frac{n}{2}+1\right)} + \frac{1}{n}\frac{C\left(\epsilon\right)}{\epsilon}, \qquad (3.33)
\end{aligned}$$

where $C(\epsilon)$ is independent of n such that

$$\begin{aligned}
C\left(\epsilon\right) &= 4K_1\sqrt{K_2}\left(2 + \frac{1}{q} + K_1\right) + 2K_1 G\left(\sqrt{\frac{\epsilon}{2K_1}}\right) \\
&\quad + \frac{4K_1^2\sigma^4}{\epsilon}G\left(\frac{\epsilon}{2\sigma^2 K_1}\right) + \frac{K_3^2}{\epsilon}G\left(\frac{\epsilon}{K_3}\right).
\end{aligned}$$

□

Chapter 4

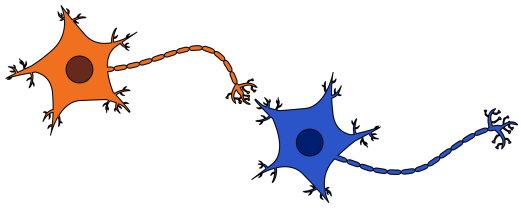
A neural network approach of complexity measure

Machine Learning (ML) is a field of artificial intelligence that consists of programming a machine to learn to perform tasks by studying examples of these tasks. From a mathematical point of view, these examples are represented by data, which the machine uses to fit a model. For example, given a set of linear functions f, $f(x) = ax + b$, the machine will produce estimators of the parameters a and b that give the best candidate model, i.e. the model that best fits the data. To do this, an optimization algorithm is programmed in the machine that will test different values of a and b until the combination that minimizes a distance or a loss function between the model and the observed points is obtained. There are many much more complicated models, such as decision trees, random forests, SVMs, neural networks, etc. Each one comes with its own optimization algorithm, Maximum margin for SVMs, CART (Classification and regression trees) algorithm for decision trees, or Gradient descent for neural networks.

Deep learning (DL) is a branch of ML, in which instead of developing one of the models mentioned above, one chains several layers of neurons which can be represented by a simple model (most of them can be seen as GLM). The main principles remain exactly the same: we provide the machine with data and use an optimization algorithm to adjust the model to these data. But this time, the model is not a simple function like $f(x) = ax + b$, but rather a network of interconnected functions, hence the term "Neural network". We will see in the following how these networks are built, and how they work for our test. What we need to know for the moment is that the deeper these networks are (i.e. the more functions they contain inside) and the more the machine can learn to perform complex tasks, such as recognizing objects, identifying a person on a photo, driving a car, and all that kind of tasks. That's why we talk about deep learning.

Origins of neural networks

The first neural networks were invented in 1943 by two mathematicians and neuroscientists named Warren McCulloch and Walter Pitts (1943) [138]. They explain in their article how they were able to program artificial neurons inspired by the functioning of biological neurons. Remember that neurons are excitable connected cells and whose role is to transmit information in our nervous system.



What Warren McCulloch and Walter Pitts tried to do was to model the functioning of a neuron, considering that a neuron could be represented by a transfer function, which takes as input signals X (say in $\mathbb{R}^k$) and returns an output (a label +1 or -1) y. Within this function, there are two main steps:

1. In an aggregation step, a weighted sum of all inputs with weights $w \in \mathbb{R}^k$ is calculated to obtain $wx = \sum w_i x_i$. The weighting coefficient represents the synaptic activity, i.e. whether the signal corresponding to x_i is excitatory when the weight is $+1$, or inhibitory when it is -1.

2. Once the 1st step is done, we go to the activation phase. We look at the result of the calculation made previously, and if it exceeds a certain threshold, then the neuron is activated and returns an output $y = 1$. Otherwise it remains at 0.

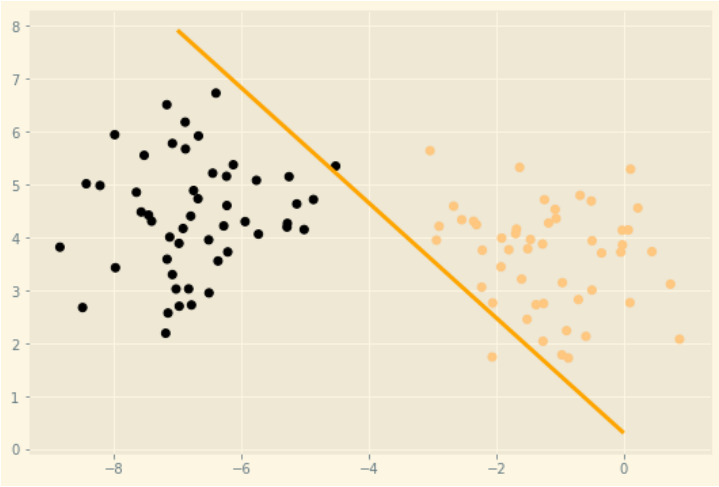
This is how Warren McCulloch and Walter Pitts succeeded in developing the first artificial neurons later renamed "Threshold Logic Unit". Their model was originally designed only to process logic inputs x_i equal 0 or 1. They demonstrated that with this model, it was possible to reproduce certain logic functions (the output), such as the *AND* gate and the *OR* gate. They also showed that by connecting several of these functions, a little bit like neurons in our brain, then it would be possible to solve any Boolean logic problem. However, their reasoning was containing several flaws, including the fact that it has no learning algorithm and that we have to find the values of the coefficients by ourselves if we want to use it in real-world applications.

About 15 years later, in 1957, an American psychologist found a way to improve this model, by proposing the first learning algorithm in the history of DL, Franck Rosenblatt, the inventor of the Perceptron [169]. To develop this algorithm, Franck was inspired by Hebb's theory which suggests that when two biological neurons are jointly excited, then they strengthen their synaptic links (i.e. they reinforce the connections between them). In neuroscience, this is called synaptic plasticity, which is what allows our brain to build its memory, learn new things or make new associations. So from this idea, Franck Rosenblatt developed a learning algorithm that consists in training an artificial neuron on reference data $\{x_i, y_i\}_{i=1..n} \in \mathbb{R} \times \{-1, 1\}$ so that it reinforces its parameters w each time an input x_i is activated at the same time as the output y_i present in these data. For this, he used the following recursive formula

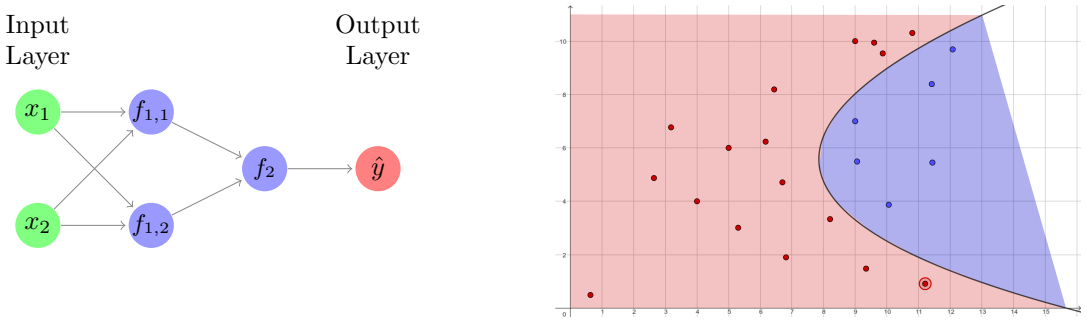
$$w(t) = w(t-1) + \alpha\,(y_i - \hat{y}_i)\,x_i \quad ; \quad \hat{y}_i = 2 \times \mathbf{1}_{w(t-1)x>0} - 1$$

As can be seen in the formula, when $\hat{y}_i = y_i$, the coefficients stay the same. But when $y_i = y_{true}$ is different from $\hat{y}_i$ then the coefficient w will move until the predictions and the reference values (the true values) are the same. Following this idea, there was excessive enthusiasm for AI. It was thought that thanks to the Perceptron, it would be possible to build machines capable of reading, speaking, walking and even having a conscience. But all this excitement collapsed a few years later when the limitations of these models were realized, partly because the perceptron is a linear model (see below). Investment in deep learning research then dropped between 1970 and 1980. But everything changed in the '80s when Geoffrey Hinton, one of the fathers of DL, developed the Multilayer Perceptron (1986), the first real artificial neural network.

For a single-layer Perceptron, we get a linear boundary that separates two classes, and the coefficients will indicate the slope of the line separating the two groups as well as its position.



The problem is that many of the phenomena are not linear. But keeping in mind the idea of McCulloch and Pitts, "by connecting together several neurons, it is possible to solve more complex problems than with a single one". By using a two-layer network, we obtain a non-linear model which is much more interesting.



Three neurons, divided into two layers (an input layer and an output layer), define a Multilayer Perceptron. Basically, one could add as many layers and neurons as needed. The more we add, the more complex and interesting the result will be. The question now is how to train such a neural network so that it performs what it is supposed to do. In other words, how to find the values of all the parameters w and b to obtain a "good" predictive model?

The answer to this is to use a technique called Back-Propagation which consists in determining how the output of the network varies according to the parameters present in each layer of the model. To do this, we calculate a chain of gradients indicating how the output varies according to the last layer and how the last layer varies according to the second last layer, and so on, until we arrive at the very first layer of the network. This is called Back Propagation. With these gradients, we can then update the values of these parameters, so that they minimize the error between the output of the model and the expected response (the value of y_{true}). To do this, one uses a formula very close to that of Franck Rosenblatt, called Gradient Descent, which we discuss in more detail in the appendix.

In summary, to develop artificial neural networks, one repeats the following four steps in a loop after initializing the weights (generally randomly).

1. *Forward Propagation:* propagate the data from the first layer to the last one in order to produce an output $\hat{y}$.

2. *Cost Function:* calculate the difference between the value of $\hat{y}$ and the value of y_{true} that we want to obtain.

3. *Back Propagation:* measure how this cost function varies with respect to each layer of the network, starting from the last one and going back to the first one.

4. *Gradient Descent:* correct each parameter of the model with the gradient descent algorithm before looping back to the first step to restart a new training cycle.

In the '90s, the first variants of the multilayer Perceptron started to be developed. Yann LeCun invented the first Convolutional networks (1990), networks that can recognize and process images, introducing at the beginning of these networks mathematical filters called Convolution and Pooling (Y. LeCun (1989) [114] Y. LeCun et al. (1989) [113]). We will talk about them in paragraph 4.1.2. It is also during these years that we saw the appearance of the first recurrent neural networks, which are once again a variant of the multilayer Perceptron and which allow us to efficiently process time series problems such as text reading or speech recognition.

The fact that all this already existed in the 90s while the technologies we see today only emerged very recently is due to two main reasons. First, in order to work properly, a neural network must be trained on a very large amount of data, sometimes exceeding millions or even billions of data. Second, computers were not as powerful with large computing capacity as they are today (with additional GPUs). DL really took off in 2012, during a computer vision competition IMAGENET, where a team of researchers led by Geoffrey Hinton, developed a neural network (G. Hinton et al. (2012) [91] or (2017) [104]), capable of recognizing any image with better performance than any other algorithm at that time. Yann LeCun says, "Comparing a neural network to a human brain is like comparing an airplane to a bird, we may have been inspired by what we have seen in nature, but that doesn't mean that airplanes fly by flapping their wings". Behind all this, there are mathematics, linear algebra, differential calculus, and a whole well-crafted mechanics that we will review in the following. For a neural networks chronology, see history of data science[1].

Neural networks and natural language processing

With the recent availability of massive textual data on the web, a variety of model architectures and approaches have been introduced in the context of natural language processing (NLP) to address different problems, such as machine translation tools or chatbot construction, etc. In other words, Deep learning methods use multiple layers of processing to learn hierarchical representations of data (see Tom Young et al. (2018) [212] who provided comprehensive coverage of the most popular deep learning methods in NLP research today. Complementing the work of Y. Goldberg (2016) [80] which presented the basic principles for implementing neural networks to perform NLP tasks in a tutorial way.).

For many years ago, NLP problems were addressed by basic machine learning models (such as knn, svm, or logistic regression) where estimation was performed using high dimensional sparse vector representations (also known as a discrete representation), relying heavily on hand-crafted features that are often time-consuming and incomplete. But since the rise of deep neural networks, results are much better for these NLP tasks, especially via dense (continuous) vector representations (see Mikolov et al. (2013) [142][143]).

Word embedding was pioneered by Mikolov (2013) [142] who proposed the CBOW and Skip-gram models. While CBOW computes the conditional probability of a target central word knowing the surrounding context words in a window of fixed size k, the skip-gram model does the opposite, by estimating the probability of the surrounding context words given a target central word. Recall also that in an unsupervised setting, the dimension of word embeddings is determined from the accuracy of the prediction. In general, the accuracy and the dimension increase in the same direction. Therefore, to obtain a parsimonious model, the dimension chosen is the one that is minimal and gives the highest accuracy, i.e. the dimension that makes the trade-off between the smallest dimension with the highest accuracy. A brief explanation of the simple case of a CBOW model was given in the first chapter, where the context to be predicted is only the next word, a variant reproducing the bigram language model.

[1] https://timeline.historyofdatascience.com/

4.1 Neural networks

In this section, we will briefly recall the main neural network models used for natural language processing, including convolutional, recurrent neural networks, as well as LSTM or networks with encoder-decoder layers and potentially with the Attention mechanism. The practical aspects will be discussed in section 2, summarizing the performance of a series of deep learning architectures on the extracted corpus from Wikipedia for building a complexity measure and an automatic text simplifier.

Notations

Denote G a model of neural networks. Let d_{in} and d_{out} represent respectively the input and the output dimensions and d_i represent the dimension of the i-th layer.

4.1.1 Simple neural network (NN)

In order to get familiar with neural networks, we will define a typical feed-forward neural network that is (drawn in Figure 4.1) without any demonstration. For more details, see the appendix.

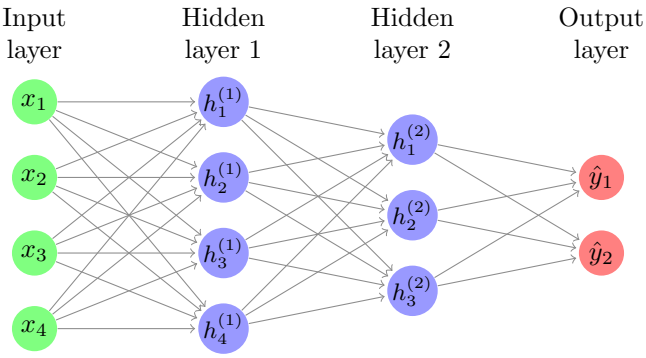


Figure 4.1: Fully connected MP2 (or MLP2 as in Y. Goldberg (2016) [80])

- **Neurons and arrows:** The neurons are represented by blue circles, with the incoming arrows representing their inputs and the outgoing arrows representing their outputs. These neurons are organized into layers that reflect the direction of information flow. As an example, let's consider the third neuron in the first hidden layer. This neuron takes in inputs $(x_1, \ldots, x_4)$ and produces an output, denoted by $h_3^{(1)}$, which is then used as an input for the neurons in the next hidden layer 2. Each arrow has a weight, indicating the importance of the incoming input information it carries, with a higher absolute weight value reflecting greater significance.

- In the figure, each neuron is connected to all of the neurons in the next layer. This is called a *fully-connected* layer or an affine layer

- **Input, hidden and output layers**

  - The input to the network is represented by the leftmost layer, which does not receive any incoming arrows.
  - The output of the network is represented by the rightmost layer, which does not have any outgoing arrows. The layers between the input and output layers are known as "hidden" layers.

– In certain cases, we can find a sigmoid shape represented inside each neuron (instead of h_j's), which indicates the use of a non-linear function (typically a $1/(1+e^{-x})$ or a tanh) that will be applied to the value of the neuron to obtain the value of the output resulting from this neuron. Some examples of these functions are given below in the paragraph about some commonly used activation functions.

- **linear and non-linear transformations** The simplest neural network is the perceptron which is a linear function of its inputs:

$$\text{For } x \in \mathbb{R}^{d_{in}}, \quad G_{\text{percetron}}(x) = xW + b$$

where $W \in \mathbb{R}^{d_{in} \times d_{out}}$ is the weight matrix, and $b \in \mathbb{R}^{d_{out}}$ is a bias term. In order to go beyond linear functions, let us introduce a non-linear hidden layer (the network in Figure above has two such layers), resulting in the 1-layer Multilayer-Perceptron[2] (MP1).

MP1, also known as a one-layer feed-forward neural network, can be expressed as:

$$G_1(x) = g_1(xW_1 + b_1)W_2 + b_2$$

where $W_1 \in \mathbb{R}^{d_{in} \times d_1}$ and $b_1 \in \mathbb{R}^{d_1}$ are a matrix and a bias term for the first linear transformation of the input. In the following $g_1, g_2, ..., g_k, ...$ are non-linear functions that is applied element-wise (they are called activation functions or non-linearities in computer science). $W_2 \in \mathbb{R}^{d_1 \times d_2}$ and $b_2 \in \mathbb{R}^{d_2}$ are the matrix and bias term for a second linear transform. The non-linear activation function g_1 plays a crucial role in the network's ability to represent complex functions. It is worth mentioning that the term "bias" used in this context refers to the constants b_i in the model and has a different meaning than the expected error of an estimator often used in statistics.

We can add additional linear transformations and non-linearities, resulting in a 2-layer MLP (the network in Figure 2 is of this form):

$$G_2(x) = (g_2(g_1(xW_1 + b_1)W_2 + b_2))W_3$$

It follows that deeper networks (with m hidden layers) can be written in the following way, using intermediary variables and resulting output y :

$$\begin{aligned}
&G_2(x) = y \text{ where} \\
&h_1 = g_1(xW_1 + b_1) \\
&\quad \vdots \\
&h_k = g_k(h_{k-1}W_k + b_k) \\
&\quad \vdots \\
&h_m = g_m(h_{m-1}W_m + b_m) \\
&y = h_m W_{m+1}
\end{aligned}$$

Linear transformations in neural networks result in layers that are commonly known as fully connected or affine layers. However, there are other types of architecture that exists. For instance, convolutional and pooling layers are particularly useful for image recognition problems, but they also have applications in language processing.

[2]Multilayer Perceptron is described in the appendix as well as the algorithm of Forward-Backward propagation

Activation functions

There are various forms that one can choose for the non-linearity function g. In the literature, some common non-linear functions used in practice include the sigmoid, the rectified linear unit (ReLU), and or the hyperbolic tangent function tanh. Currently, there is no established theory on which non-linear function should be used under specific conditions. Thus, selecting the appropriate non-linear function g for a given task is mainly an empirical inquiry. Some NLP researchers have experimented with other non-linear forms such as hard tanh, cube, and tanh-cube. On the one hand, the S-shaped sigmoid activation function $\sigma(x) = 1/(1 + e^{-x})$ transforms each value x into the range [0, 1] and is widely used. On the other hand, the Rectifier activation function (Glorot, Bordes and Bengio, 2011), also known as the rectified linear unit, is a straightforward activation function that is easy to work with: it is known to give good results in practice. It is defined by

$$\forall x \in \mathbb{R}, \quad ReLU(x) = \max(0, x) = \begin{cases} 0 & \text{if } x < 0, \\ x & \text{otherwise.} \end{cases}$$

A common transformation of the output layer vector is the softmax function given by :

$$x = (x_1, \ldots, x_k) \in \mathbb{R}^k$$
$$\text{softmax}(x) = \left(\frac{\mathrm{e}^{x_i}}{\sum_{j=1}^{k} \mathrm{e}^{x_j}} \right)_{i \in \{1,\ldots,k\}}$$

This transformation, also known as logistic transformation in the statistical literature, yields a vector of non-negative real numbers summing to one. The output is thus a discrete probability distribution over k possible outcomes. The softmax transformation is thus adapted in modeling a probability distribution over all possible output classes: in NLP, and particularly in our framework, where we want to predict the probability of some classes or words, this is generally the preferred one. It will be used in conjunction with a probabilistic training objective or loss such as the cross-entropy. From a statistical point of view, this will generate at each layer a multinomial logistic regression model and the corresponding maximum-entropy classifier, see Chapter 2.

Input and embedding Layer

So far, the discussion has treated x as an arbitrary vector, without considering its source. However, in NLP applications, x is typically composed of various embedding vectors. To be more specific about the origin of x, we can include it in the network's definition by introducing a function f that maps core features to an input vector. So there are two possibilities:

- We can manually define the important features and provide the words (already transformed into feature vectors) as inputs, denoted as x_i, to the network. The resulting feature vector is typically of high dimension.
- Alternatively, we can provide only the sparse vector corresponding to each word, where all elements are 0 except for the position of 1, indicating which word from the vocabulary/dictionary is being fed into the network. In this case, the first layer of the network is responsible for feature extraction, transforming the words into a dense and low-dimensional feature vector.

Loss Functions

In both neural networks and linear classifiers, a loss function $L(\hat{y}, y)$ is used to measure how well the model is predicting the correct output. The loss function assigns a score to the model's prediction $\hat{y}$ compared to the actual output y. The goal is to minimize this score across all training examples by adjusting the network's parameters (such as the matrices and biases). This is done using an optimization

algorithm that relies on computing the gradients of the loss function. To make optimization easier, the loss function should be differentiable so that its gradients can be computed efficiently.

Hinge (binary) For a binary classification problem, the neural network outputs a single scalar value $\hat{y}$, while the intended output y belongs to $+1, -1$. The classification rule is determined by taking the sign of the output, which means that the predicted output $\hat{y}$ is assigned to the positive class if it is greater than or equal to zero, and to the negative class otherwise. A classification is considered correct if y and $\hat{y}$ have the same sign. This means that a positive example is correctly classified if $\hat{y} \geq 0$, while a negative example is correctly classified if $\hat{y} < 0$. The hinge loss, also known as margin loss or SVM loss, is a popular loss function for binary classification problems. It is defined as:

$$L(\hat{y}, y) = \max(0, 1 - y \times \hat{y})$$

The hinge loss has a value of zero when y and $\hat{y}$ have the same sign and $|\hat{y}| \geq 1$. In this case, the margin between the predicted output and the true output is at least 1, which is desirable in classification problems. On the other hand, when y and $\hat{y}$ have opposite signs or $|\hat{y}| < 1$, the hinge loss increases linearly with the margin between the predicted output and the true output. This encourages the neural network to correctly classify examples with a margin of at least 1.

Hinge (multiclass) The multiclass hinge loss is an extension of the hinge loss for binary classification problems. It was introduced by Crammer and Singer in 2002. In the multiclass setting, the network's output is a vector $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_K)$, where K is the number of classes. The intended output y is a one-hot vector representing the correct output class.

The correct class is denoted by $t = \arg\max_i (y_i)$, and the second highest scoring class is denoted by $k = \arg\max_{i \neq t} (\hat{y}_i)$. The hinge loss is given by

$$L(\hat{y}, y) = \max(0, 1 - (\hat{y}_t - \hat{y}_k))$$

The loss is 0 when the score for the correct class $\hat{y}_t$ is greater than or equal to the score for the highest scoring class $\hat{y}_k$ by a margin of 1 or more. Otherwise, the loss is linear with respect to the difference between the scores, $\hat{y}_t - \hat{y}_k$.

The multiclass hinge loss is commonly used in Support Vector Machine (SVM) based methods for multiclass classification. It is a convex function that can be optimized using standard convex optimization techniques. Like the binary hinge loss, it is a margin-based loss, which means that it tries to maximize the margin between the scores for the correct class and the highest scoring class. However one generally prefers to use the Log Loss.

Log loss The log loss is a variation of the hinge loss. It can be seen as a "soft" version of the hinge loss with an infinite margin (LeCun et al., 2006). It is given by

$$L(\hat{y}, y) = \log(1 + \exp(-(\hat{y}_t - \hat{y}_k))$$

Categorical cross-entropy loss If one want to interpret the scores of a neural network probabilistically, one generally use the categorical cross-entropy loss, also called negative log-likelihood. The categorical cross-entropy loss measures the dissimilarity between the true label distribution y and the predicted label distribution $\hat{y}$. It is given by the formula:

$$L(\hat{y}_i, y) = -\sum_i y_i \log(\hat{y}_i)$$

In this case, $y = (y_1, \ldots, y_K)$ is a vector representing the true multinomial distribution over the labels $1, \ldots, K$, and $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_K)$ is the network's output transformed by the softmax activation function, and representing the estimated class membership conditional distribution, i.e., $\hat{y}_i = \hat{\mathbb{P}}(y = i|x)$. These quantities may be obtained for instance with the methods presented in chapter 2 or using neural network as will be done later.

The categorical cross-entropy loss ensures that the network's output probabilities match the true probabilities as closely as possible. In other words, it tries to minimize the distance between the predicted

probability distribution and the true probability distribution. When the predicted probabilities are close to the true probabilities, the loss is small, and when the predicted probabilities are far from the true probabilities, the loss is large.

Different tasks

Supervised Task-specific Pre-training

A common issue in machine learning is the scarcity of labeled data for a specific task. A possible solution involves using an auxiliary task that has more labeled data, to help improve performance on the target task.

For instance, suppose we are interested in task A, which requires labeled data for syntactic parsing, but we only have a limited amount of it. In contrast, we have a significant amount of labeled data for an auxiliary task B, for instance, part-of-speech tagging. To take advantage of the larger labeled dataset for task B, we can pre-train our word vectors to perform well as predictors for task B. We can then use these pre-trained vectors for task A, where they can be fine-tuned to further optimize their performance on that task.

There are several ways we can use the pre-trained word vectors for task A. We can either keep them fixed and use them as input for training the model, or we can adjust them to better fit the requirements of task A. Alternatively, we can train the model jointly for both tasks A and B.

Overall, this approach is an example of transfer learning (see for instance Bertail et al. 2022 and some references therein), where knowledge gained from one task is used to enhance performance on another task. By leveraging the larger labeled dataset for task B, we can learn better representations of language that can be applied to task A, even when labeled data is limited.

Unsupervised Pre-training The vast majority of the time, we don't have any related auxiliary task with enough annotated data available. In these situations, one uses "unsupervised" methods that can be trained on vast quantities of unannotated text. The methods for training the word vectors are similar to those used in supervised learning, but instead of supervision for the task that we're interested in, one creates an enormous number of practically unlimited supervised training examples from raw texts. The key concept behind unsupervised methods is that we want embedding vectors for "similar" words to be similar: this relies on the distributional hypothesis (Harris (1954) [89]), which suggests that words are similar if they appear in similar contexts. The various techniques all create supervised training examples in which the objective is to either predict the word based on its context or predict the context based on the word.

4.1.2 Convolutional neural network (CNN)

Word embeddings have been very popular for their ability to represent words in a distributed space. Then, the necessity to build effective feature functions that allow to extract more advanced features has increased, compared to the classical basic feature building as with n-grams for example. This could potentially improve several NLP tasks such as sentiment analysis, summarization or machine translation, etc. One of the first deep neural networks that researchers have considered is CNN, for its performance in computer vision tasks (see Krizhevsky et al. (2017) [104]).

Using CNNs for sentence modeling was initiated by Collobert [48] in multiple NLP tasks, to predict the POS-tags, chunks, semantic similarity between words etc. CNN has shown an ability to extract important n-gram features from the input sentence, in order to provide a latent semantic representation of the sentence that is relevant to the underlying task [5]. This has led to increased interest in using CNN-based networks for NLP tasks in the literature.

A simple basic CNN

Consider an input sentence s having n words $x_1 \dots x_n$, where x_i is the word embedding for the ith word in the sentence (which means that we denote by w_i the vector representation and not a string or

alphanumeric object). Now the sentence s can be represented by a matrix $X \in \mathbb{R}^{n \times d}$ if we consider that each $x_i \in \mathbb{R}^d$ with d representing the dimension of the word embedding.

Denote the sequence $w_{i:j}$ a subset of consecutive words from the original sentence i.e.

$$x_{i:j} = x_i \dots x_j \quad \text{for all } 1 \leq i \leq j \leq n,$$

the sequence that concatenates the inputs x_i, x_{i+1},... x_{j-1}, x_j.

Then convolution is performed on those sequences. Let $W \in \mathbb{R}^{h \times d}$ be a filter applied to a window of h words to produce a feature c_i^h as follows

$$c_i^h = f\left(x_{i:i+h-1} W' + b\right)$$

where f is a non-linear activation function and $b \in \mathbb{R}$ is a bias term.

The filter k is applied to all possible windows using the same weights to create the feature map c^h given by

$$c^h = \left\{c_1^h, \dots, c_{n-h+1}^h\right\}$$

Filters (also called kernels) of different widths slide through the entire word embedding matrix. Each filter extracts a specific feature from the n-gram. Each convolution layer is often followed by a max-pooling $\hat{c}^h = \max_i \left(c_i^h\right)$ strategy which selects a part of the input, by applying the maximum for each filter. This technique has several benefits, two of the main reasons being that it provides an output of a fixed-size on the one hand, which is required for classification for example. Thus, whatever the size of the filter, max-pooling will always transform the input into a fixed-size output. On the other hand, it enables reducing the dimension while preserving the most important n-gram features in the whole sentence. This is done in a translation-invariant way, where each filter is now able to extract a particular feature from any position in the sentence and add it to the final sentence representation.

Concerning initialization, the network can be initialized with either random values or with pre-trained values on large unlabeled corpora. It turns out that initialization with pre-trained values leads to better performance, especially when the amount of labeled data is limited. These sequential convolutions improve sentence representation by capturing richer semantic information, leading to a global summary of the input sentence features.

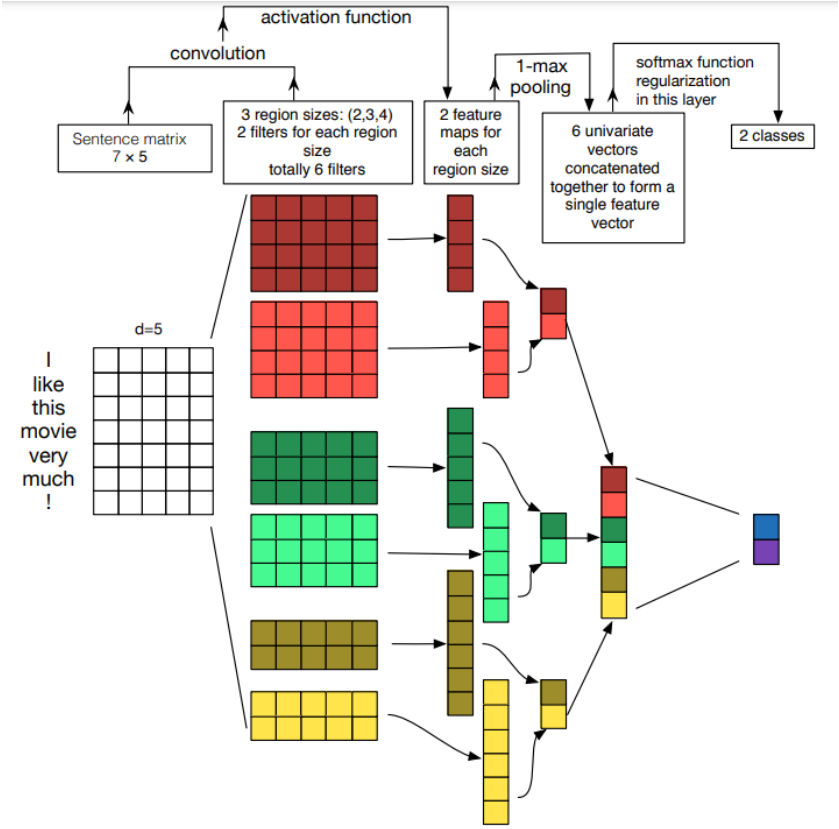


Figure 4.2: Illustration of a CNN architecture for sentence classification (see Zhang and Wallace (2015) [216])

4.1.3 Recurrent neural network (RNN)

Recurrent neural networks (RNNs) are a type of artificial neural network that excels in sequential data processing. Unlike traditional feed-forward neural networks, RNNs have connections that allow information to flow in loops, enabling them to capture and utilize context information from previous steps in the sequence. This makes RNNs particularly effective for tasks involving sequences, such as natural language processing and speech recognition. RNNs use the principle of sequentially processing information. The term "recurrent" refers to the fact that the model performs the same task on each instance of the sequence so that the output depends on the previous calculations and results. In general, a vector of fixed size is produced to represent a sequence, feeding each recurrent unit with the words that constitute this sequence one by one in succession. Thus, RNNs have a kind of memory that stores previous computations in order to use it for the current processing. This approach is clearly well suited to NLP tasks such as language models or machine translation.

The semantic meaning of words in a language is often influenced by the preceding words in a sentence. For instance, consider the distinction between "dog" and "hot dog." RNNs are specifically designed to handle such contextual dependencies in language and other sequence modeling tasks. This aspect has proven to be a compelling reason for researchers to favor RNNs over Convolutional neural networks (CNNs) in these domains (see Zhang and Wallace (2015) [216]). Additionally, RNNs possess an advantage in handling variable-length text, encompassing long sentences, paragraphs, and even entire documents. Unlike CNNs, RNNs offer flexible computational steps that enhance their modeling capabilities, enabling them to capture unbounded contextual information. This ability to handle input of arbitrary length has emerged as a prominent feature in notable works utilizing RNNs (see LeCun et

al. 2015 [112]).

One of the key strengths of RNNs lies in their ability to model variable-length text, such as sentences, paragraphs, and documents. Unlike Convolutional neural networks (CNNs), which have fixed computational steps, RNNs can adapt their computation to handle sequences of different lengths. This flexibility allows RNNs to capture long-range dependencies and unbounded context, making them well-suited for tasks that require understanding the sequential nature of data.

The mathematical representation of a basic Recurrent Neural Network (RNN) can be summarized as follows:

$$
\begin{aligned}
h_t &= \tanh(W \cdot [h_{t-1}, x_t] + b_h) \\
y_t &= \text{softmax}(U \cdot h_t + b_y)
\end{aligned}
$$

Here, h_t represents the hidden state at time step t, and x_t represents the input at time step t. The matrices W and U are weight matrices, while b_h and b_y are bias terms.

The dot symbol $(\cdot)$ represents matrix multiplication. The vectors $[h_{t-1}, x_t]$ represent the concatenation of the previous hidden state h_{t-1} and the current input x_t. This concatenated vector is then multiplied by the weight matrix W and added to the bias term b_h. The result is passed through the hyperbolic tangent function (tanh) to compute the current hidden state h_t.

The output y_t is obtained by multiplying the hidden state h_t by the weight matrix U and adding the bias term b_y. The softmax function is then applied to obtain the final output probabilities.

4.1.4 Long Short Term Memory neural network (LSTM)

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture that addresses the vanishing and exploding gradient problem and enables better long-term dependency modeling. LSTMs are designed to retain information over longer sequences, making them particularly effective for tasks that involve capturing and understanding temporal patterns.

The key innovation in LSTM networks is the addition of memory cells, which allow the network to selectively remember or forget information. This memory mechanism enables LSTMs to learn and retain relevant information over extended sequences, mitigating the issues of vanishing or exploding gradients that can hinder traditional RNNs. By effectively managing the flow of information through a series of gates, including input, forget, and output gates, LSTMs can store and retrieve information over long periods, making them well-suited for tasks such as speech recognition, language translation, and sentiment analysis.

The mathematical representation of an LSTM cell can be summarized as follows:

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\
C_t &= f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
h_t &= o_t \cdot \tanh(C_t)
\end{aligned}
$$

Here, f_t represents the forget gate, i_t represents the input gate, and $\tilde{C}_t$ represents the candidate cell state. C_t denotes the updated cell state, and o_t represents the output gate. Finally, h_t represents the hidden state/output of the LSTM cell.

The dot symbol $(\cdot)$ denotes matrix multiplication. The matrices W_f, W_i, W_C, W_o are weight matrices, and the vectors $[h_{t-1}, x_t]$ represent the concatenation of the previous hidden state h_{t-1} and the current input x_t. The bias terms b_f, b_i, b_C, b_o are added to the weighted sums.

The sigmoid function (σ) is applied element-wise, while the hyperbolic tangent function (tanh) is also applied element-wise. These activation functions introduce non-linearity to the LSTM computations.

4.1.5 Encoder-Decoders and Transformers

A **transformer** is a deep learning model used in natural language processing (NLP) and computer vision (CV) that applies the self-attention mechanism, allowing for more parallelization and reducing training times compared to recurrent neural networks (RNNs). It processes the entire input all at once, and the attention mechanism provides context for any position in the input sequence. Transformers are replacing RNN models like LSTM and are more amenable to parallelization, allowing training on larger datasets. Pretrained systems like BERT and GPT, which were trained with large language datasets, are developed and can be fine-tuned for specific tasks. OpenAI introduced the first GPT system in 2018, which employed an unsupervised generative "pre-training" stage and a supervised discriminative "fine-tuning" stage.

Natural language understanding encompasses a broad range of tasks, including textual entailment, question answering, semantic similarity assessment, and document classification. Despite the abundance of large unlabeled text corpora, there is a scarcity of labeled data for these specific tasks, which poses a challenge for discriminatively trained models to perform adequately. However, Radford (2018) [163] shows that significant progress can be made by pre-training a language model generatively on a diverse corpus of unlabeled text, followed by discriminative fine-tuning on each task. Their method differs from previous approaches in that they employ task-specific input transformations during fine-tuning to achieve effective transfer with minimal changes to the model architecture. They illustrate the effectiveness of our approach on a wide variety of benchmarks for natural language understanding. Their paper describes the success of a general task-agnostic model compared to discriminatively trained models that are tailored to specific tasks. The general model performs significantly better than the specialized models in 9 out of 12 tasks, achieving absolute improvements ranging from 1.5% to 8.9% in tasks such as textual entailment, question answering, and commonsense reasoning.

According to Young (2018) [212], the use of pre-trained language models for various natural language tasks has recently gained significant attention. Language modeling is a common pre-training objective as it involves learning complex linguistic characteristics. Generative pre-training and discriminative fine-tuning procedures are desirable as they do not require manual labeling. Two popular pre-trained models are OpenAI-GPT and BERT, both utilizing the transformer network for pre-training a language model. BERT uses different pre-training tasks for language modeling, including predicting masked words and predicting the next sentence. This approach has been shown to outperform traditional techniques on NLP tasks such as Question-Answering (QA) and Natural Language Inference. These pre-trained models promise better quality representations for words and provide a headstart for downstream tasks through transfer learning. The future preference for such models over traditional variants in the NLP community remains to be seen.

Arora (2017) [6] shows that Unsupervised methods can beat state-of-the-arts supervised models. Their paper (Arora (2017) [6]) discusses methods for generating semantic embeddings of longer texts, such as sentences and paragraphs, and how simpler methods involving mild retraining of word embeddings and basic linear regression outperform complicated methods in out-of-domain settings. The paper proposes a completely unsupervised sentence embedding method as a baseline, which involves using word embeddings computed on unlabeled corpora like Wikipedia, representing the sentence by a weighted average of the word vectors, and modifying them slightly using PCA/SVD. This method improves performance in textual similarity tasks by 10-30% and even beats sophisticated supervised methods. The paper also provides a theoretical explanation for the success of this unsupervised method using a latent variable generative model for sentences.

The traditional models for sequence transduction use complex neural networks that include an encoder, a decoder, and an attention mechanism connecting them (see the paragraph 4.1.5). Vaswani et al. (2017) [199] propose in their paper a new architecture called the Transformer that solely relies on attention mechanisms, eliminating the need for recurrence and convolution. The Transformer is faster, more parallelizable, and produces higher-quality results than the traditional models. On the English-to-German translation task, the Transformer achieved 28.4 BLEU[3] (bilingual evaluation understudy), outperforming the existing best results by over 2 BLEU. On the English-to-French task, the Transformer established a new state-of-the-art BLEU score of 41.0, with significantly less training time and resources than the best models in the literature.

Liu et al. (2018) [126] used a variant of Vaswani (2017) multi-layer Transformer decoder for the language model. They used Decoders only (without encoders). In their study, the authors propose a method for generating English Wikipedia articles by treating it as a multi-document summarization task. The proposed approach involves using extractive summarization to identify important information and a neural abstractive model to generate the article. The abstractive model utilizes a decoder-only architecture that can attend to long sequences, enabling it to generate coherent paragraphs and articles. The method is evaluated based on perplexity, ROUGE[4] scores (Recall-Oriented Understudy for Gisting Evaluation), and human evaluations and is shown to be effective at extracting relevant factual information when given reference documents.

Attention mechanism

In artificial neural networks, attention is a technique used to imitate cognitive attention by enhancing important parts of input data while reducing the significance of other parts. This helps the network focus more on crucial data, even if it's small. Learning which parts of data are more important than others depends on the context and is trained by gradient descent. The term "Attention" in the English language refers to the act of directing one's focus toward something and giving it greater consideration. In Deep Learning, the Attention mechanism is inspired by this concept, where it selectively emphasizes specific aspects when processing data.

The attention mechanism was introduced by Bahdanau et al. in (2014) [7] to address the bottleneck problem that arises when using a fixed-length encoding vector. This can limit the decoder's access to input information, particularly for long and complex sequences where their representation dimensionality would be forced to be the same as shorter or simpler sequences.

4.2 Simplification measure

The dataset used will always be the Wikipedia one, the complex version (standard English), and the simple version.

The following subsection describes the choice of the database or corpus that will be used to train the binary classification models of an input text through the prediction of its class (simple or complex) as an output. This section gives also the main steps of the extraction process as well as the major problems encountered during these steps. The second subsection discusses the results obtained, in particular using deep neural networks and penalized maximum entropy methods.

4.2.1 Extraction

Several issues were raised while extracting the data from the Wikidepia corpus :

[3]BLEU (bilingual evaluation understudy) is an algorithm for evaluating the quality of text that has been mechanically translated from one natural language to another (see Papineni et al. (2002) [153])

[4]ROUGE is a set of metrics and a software package used for evaluating automatic summarization and machine translation software in natural language processing (see Chin-Yew Lin (2004) [125]).

- What are the best objects to extract, to build a good model or to improve it?
  - Texts, paragraphs, sentences? Actually, it depends on the task we want to perform. For translation, the best is to process sequences of words. For text generating, the best results are obtained with sequences of characters (called stems, for instance as used in chatGPT). However, for text classification, the best results are obtained using the whole text including all sentences.
  - Specific texts on a theme, a category or a field, or general texts without distinction? We found out that focus on a specific thematic context, we always get better performance.
- What is the impact of sentence sizes on classification and simplification? Notice that the sizes of the sentences will certainly be very different between the complex and simple versions: this may cause problems to align texts. To build a measure of complexity, we found out that the sentence size is not a problem, however, too big differences in sentence sizes may cause problems to simplify a text (just as for automatic translation).
- How to control the extraction so that there are no errors, without having to look at the extracted texts one by one. See the indicators below.
- How to transform extracted objects into vectors? In a discrete way into high dimensional binary vectors, or in a continuous way using word2vec (or *doc2vec, count vectorizer, tfidf-vectorizer* etc.) We will explain this transformation in paragraph 4.2.2.
- Which models should be used and implemented? In our case, we will mainly use the penalized maximum entropy principle introduced in Chapter 2, as well as deep neural networks.

It is important to note that the goal is actually to use a corpus in order to build a model and not to study the corpus itself. This means that all results will be relative to the corpus used. Thus, using the same methods on a different corpus would surely lead to different results.

Text Extraction for Simplified text generator
For the extraction process, we consider indicators, which are key parameters for the classification and simplification tasks.

- **Indicator 0:** the choice of texts to be extracted. It can be as mentioned before, whole texts, sentences, or part of sentences.
- **Indicator 1:** The number of texts to be extracted.
  The ideal scenario would be to extract all the texts because more data means better results with greater precision. Except that the machine has limited memory, especially when one transforms the texts into lists of sentences, each sentence into a list of words, and then transforms them into numerical data. All these tasks can be very expensive in terms of memory and calculation, especially if the number of extracted texts is very large.
- **Indicator 2:** Missing data.
  We want to avoid extracting texts that have missing data. For this, several filters have been implemented. For example, we consider a filter that looks at the size of the extracted texts; if the size of the text does not exceed a minimum number of characters, it will be ignored. Other filters that ignore texts that contain typical error messages in the Wikipedia dumps, such as "...Pages for logged..." or "Other reasons this message may be displayed:..."

  Example of pre-extraction check We have many tools that allow checking the extracted texts, one of them is the wordcloud that allows to display a cloud of the most frequent words, with each word having a proportional size to its frequency. As we can see, the second cloud 4.4 contains

many times the word displayed, which indicates that there are many texts that were not extracted in the right way.

Figure 4.3: Extraction without errors

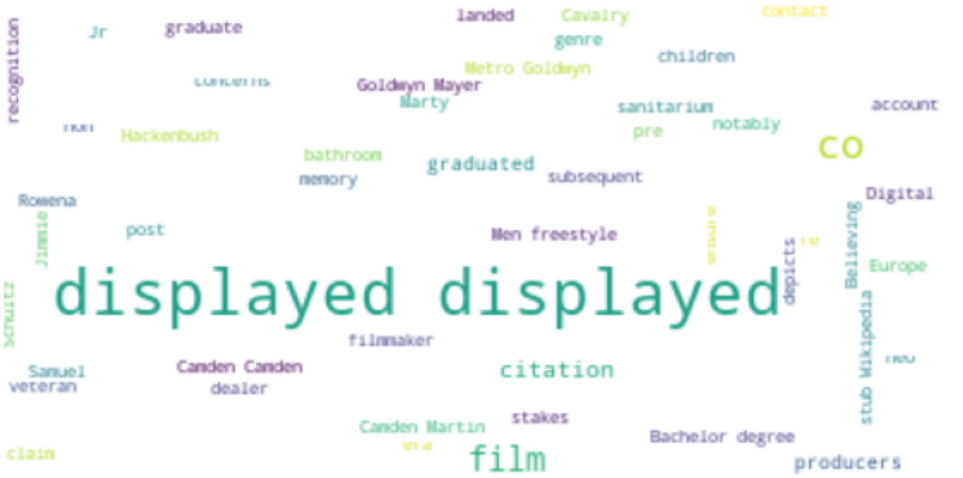


Figure 4.4: Extraction with errors

This is only a visual check, but we need more processing to avoid those extraction issues.

- **Indicator 3:** Unbalanced bitexts:
  Sometimes one of the two versions is much more under-documented than the other one. In general, standard English texts are much richer, longer, and contain more information, i.e. more data (more words, more sentences..) than in the simplified version. Since automatic text simplification is a task that consists in reducing complexity and not really size, we will avoid bi-texts with two different sizes (which are more adapted for summarizing). Notice that summarizing is a different task from simplifying. Thus we consider only texts that have more or less the same size in both versions. A filter of the type $0.5 \leq \frac{size(\text{simple text})}{size(\text{complex text})} \leq 1.5$ is used for that.

- **Indicator 4:** Polluted and noisy texts
  Often on Wikipedia, texts contain noise, such as numbers in square brackets indicating different

definitions. Sometimes texts contain forgotten spaces between a dot and the next word, which could confuse the program (that splits the sentence into words) by making it understand that the following sequence "*word1, word2*" is only one word when in reality they are three words "*word1*" and "," and "*word2*" (since punctuation is considered a word in its own right). Here are some other "noises" we can find: long empty messages (having a valid size and no error message), foreign languages, simple text's title (itself simplified) completely different from the standard English one, and so on.

Example of post-extraction check In the following table, we can see that the sentences in the first table are not cleaned. Notice the numbers inside brackets for instance on row number 3 or 4, indicating the fourth description of an article in Wikipedia. There are also some python's special string characters such as "\n" on line 19216 and so on...

	text	category	length
0	Cynthia is a 1947 American comedy drama movie ...	simple	27
1	It stars Elizabeth Taylor, George Murphy, S. ...	simple	22
2	0 is an Indian science fiction movie written a...	simple	22
3	[4] The movie serves as a spiritual successor ...	simple	31
4	[5] With an estimated budget of ₹570 crore it ...	simple	18
...	...	...	...
19213	For several years, Wright teamed with Chicago ...	complex	25
19214	Born in Orlando, Florida, he toured as a tap d...	complex	26
19215	Wright died of cancer at his home in Chicago i...	complex	11
19216	[1]\n\nThis article about a United States film...	complex	16

	text	category	length
0	Cynthia is a 1947 American comedy drama movie ...	simple	27
1	It stars Elizabeth Taylor, George Murphy, S. Z...	simple	22
2	0 is an Indian science fiction movie written a...	simple	22
3	The movie serves as a spiritual successor to ...	simple	31
4	With an estimated budget of ₹570 crore it is ...	simple	18
...	...	...	...
19213	For several years, Wright teamed with Chicago ...	complex	25
19214	Born in Orlando, Florida, he toured as a tap d...	complex	26
19215	Wright died of cancer at his home in Chicago i...	complex	11
19216	This article about a United States film actor...	complex	16

Remark: In order to avoid aligning texts by sequences, which is an open field of research in NLP at the moment, we will make a posteriori selection (after extraction) of texts having only a reasonable number of words. Indeed, the majority of translators require bitexts that are aligned in terms of sequences, i.e. each sequence in the source corpus has a unique correspondent in the target corpus.

Part of sentences (phrases) are mostly used in practice by online translators; however, DeepL performs translation at the level of the aligned sequences extracted from the linguee[5] website.

Text Extraction for Complexity measure classifier Several indicators among those previously mentioned for the automatic text simplification (the generator model) will be reused for the extraction of the database to train the complexity measure. Neither Indicators 0 and 3 nor aligned bitext are necessary. On the other hand, in practice, the accuracy of the models is much better with texts coming from the same category than with texts coming from different categories. All other indicators are important and should be taken into account. To save time and computational resources, we decided to extract aligned bitext to train the neural network (to construct the text generator), and then to use a random sub-extraction to fit the complexity measure model.

4.2.2 Results

In the first paragraph, we describe the results obtained by using Penalized Maximum entropy models as in Chapter 2 combined with a feature extraction stage. In the second paragraph, we present different deep neural network architectures and apply them to our problem.

Without using deep neural networks Each of the extracted texts can contain multiple sentences, which in turn contain multiple words. Thus, the construction of features is done in several steps:

- Each text is an observation associated with a label according to the corpus from which it comes (0 if it is simple and 1 if it is complex).
- Each text is first transformed into a context vector whose components are the n-grams of the text (unique words, bigrams, and trigrams), the sentence length (in terms of the number of words), and the average word length (in terms of the number of characters).
- Each context vector is then transformed into a feature vector in the same spirit as Chapter 2, meaning that the features will check if any of the components of the context coincide with the same label multiple times (more than 10 times).
- Thus, we obtain several pairs (feature vector, label) where each feature vector is a representation of the observed text by a binary vector with 1s where the condition is met and 0s where it is not. It also contains some non binary features such as sentence-length information for example.

Let's consider the following two texts as an example, the first observed in the simple corpus and the second in the complex corpus.

$t_1 =$ "An alphabet (otherwise known as a word system) is a way to write words and other ..."

$t_2 =$ "An alphabet is a standardized set of basic written graphemes (called letters) repr ..."

After selecting the features that appear more than ten times, here is the part/block of the feature vector that looks at the (word, label) pairs, say f_1.

$$Unigram = \{(\text{a}, 0), (\text{a}, 1), (\text{is}, 0), (\text{is}, 1), (\text{the}, 0), (\text{the}, 1), ..., (\text{standardized}, 1), (\text{yes}, 0), (\text{yes}, 1)\}$$

$$f_1(t_1, 0) = (1, 0, 1, 0, 0, 0, \ldots, 0, 0)$$
$$f_1(t_1, 1) = (0, 1, 0, 1, 0, 0, \ldots, 0, 0, 0)$$
$$f_1(t_2, 1) = (0, 1, 0, 1, 0, 0, \ldots, 1, 0, 0)$$

There is also a block that looks to the pairs (bigram, label) say f_2:

$$Bigram = \{(\text{is the}, 0), (\text{is the}, 1), (\text{is a}, 0), (\text{is a}, 1), (\text{a standardized}, 0), (\text{are the}, 1), \ldots, (\text{yes it}, 0)\}$$

[5] https://www.linguee.fr/

$$f_2(t_1, 0) = (0, 0, 1, 0, 0, 0, \ldots, 0, 0)$$

$$f_2(t_2, 1) = (0, 0, 0, 1, 1, 0, \ldots, 0)$$

Finally, all these blocks are concatenated to obtain two feature vectors for each sentence, one for the pair (sentence, label=0), and the other for the pair (sentence, label=1).

$$f(t_1, 0) = (f_1(t_1, 0), f_2(t_1, 0), \ldots, f_K(t_1, 0))$$

$$f(t_1, 1) = (f_1(t_1, 1), f_2(t_1, 1), \ldots, f_K(t_1, 1))$$

Once the features are constructed, we proceed in the same way as for POS-tagging, meaning that we estimate the conditional probabilities $\mathbb{P}(y|x)$. Then, the class y that is assigned to x is the one that maximizes these conditional probabilities:

$$\forall x, \quad y = \begin{cases} 1 & \text{if } \mathbb{P}(1|x) > 1/2 \\ 0 & \text{else} \end{cases} \quad \text{where } \forall y \in \{1, 2\}, \ \mathbb{P}(y|x) = \frac{\mathrm{e}^{\lambda' f(x,y)}}{\mathrm{e}^{\lambda' f(x,1)} + \mathrm{e}^{\lambda' f(x,1)}}$$

These probabilities are estimated as follows

$$\hat{\mathbb{P}}(y|x) = \frac{\mathrm{e}^{\hat{\lambda}' f(x,y)}}{\mathrm{e}^{\hat{\lambda}' f(x,0)} + \mathrm{e}^{\hat{\lambda}' f(x,1)}}$$

where $\hat{\lambda}$ is obtained by maximizing the likelihood. Notice that we can also directly use the following expression (as proposed in Chapter 2):

$$\hat{\lambda} = -(\bar{f}_n - \hat{\mu}_N)' \left(S_n^2 + \rho_n I_q\right)^{-2} \text{ with } S_n^2 = \frac{1}{n} \sum_i^n (f(x_i, y_i) - \hat{\mu}_N)(f(x_i, y_i) - \hat{\mu}_N)'$$

where $\hat{\mu}_N = \frac{1}{N} \sum_{i=1}^N f(x_i, y_i)$ is obtained from the entire initial dataset of size N (before splitting it into training and test set), and $\bar{f}_n = \frac{1}{n} \sum_{i=1}^n f(x_i, y_i)$ the empirical mean obtained on the training dataset of size n. We used the maximum likelihood method to avoid the following problem: sometimes the empirical mean of the training sample $\bar{f}_n$ coincides with the mean of the global sample $\hat{\mu}_N$, which amounts to considering a uniform model as follows:

$$\forall x, \quad \hat{\mathbb{P}}(0|x) = \hat{\mathbb{P}}(1|x) = \frac{1}{2}$$

Using deep neural networks

All the following steps will be considered as a trial: Extraction → processing → modelling. Each of these steps can be calibrated or improved in order to obtain better results. Thus, the following paragraph presents a first trial, with all the descriptive statistics that go with it from extraction up to the modeling. Then the paragraph "intermediate results" summarises all the intermediate tests and the results obtained in a table with many cases and scenarios for each of the three steps of each trial. The last paragraph gives the final result and an interpretation of it.

First result

We first extract 21945 texts (each text here is a sentence): 9829 simple texts and 12117 complex texts. See the first and last observations in the table 4.5 below.

	text	category
0	The Hertzsprung-Russell diagram is a graph of ...	simple
1	It shows the relation between stars' luminosit...	simple
2	These diagrams are not pictures or maps of the...	simple
3	Rather, Hertzsprung-Russell diagrams plot each...	simple
4	Hertzsprung-Russell diagrams are also called H...	simple
...	...	...
21941	The male equivalent of an aunt is an uncle, an...	complex
21942	A female cousin of one's parent is often, alth...	complex
21943	Due to the positive image of an old but wise a...	complex
21944	Children's TV hosts using "aunt" as their nick...	complex
21945	New Zealand radio presenter Aunt Daisy is anot...	complex

21946 rows × 2 columns

Figure 4.5: First extraction

The number of sentences in each category is represented in the graph 4.6 below.

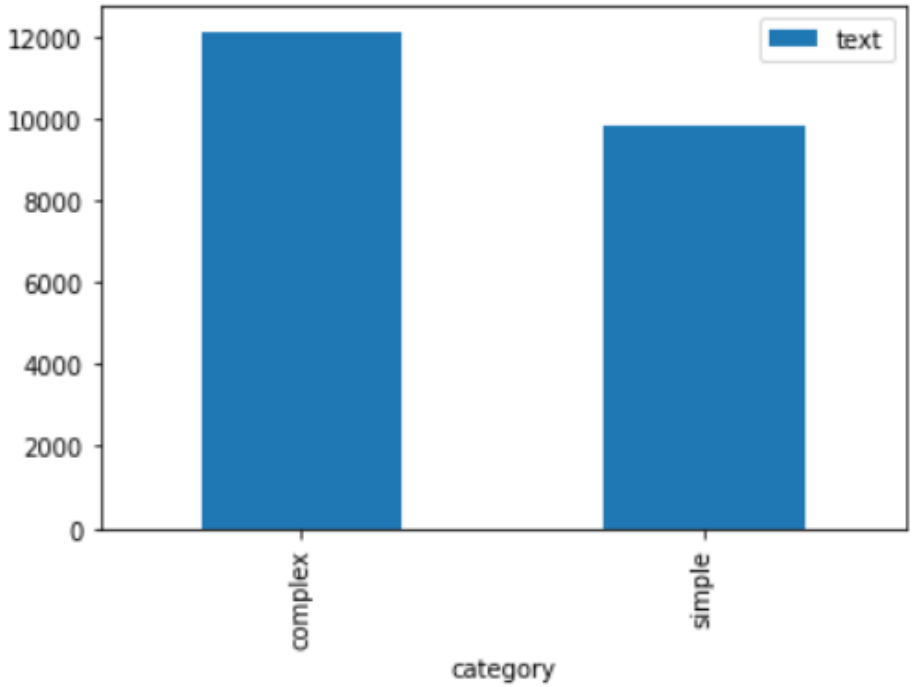


Figure 4.6: Number of texts

The two graphics below 4.7 and 4.8 represent the distributions of the lengths of sentences (in terms of the number of words). Simple version first and then the complex version.

	length
count	9829.0
mean	14.93
std	8.73
min	1.0
25%	9.0
50%	13.0
75%	19.0
max	128.0

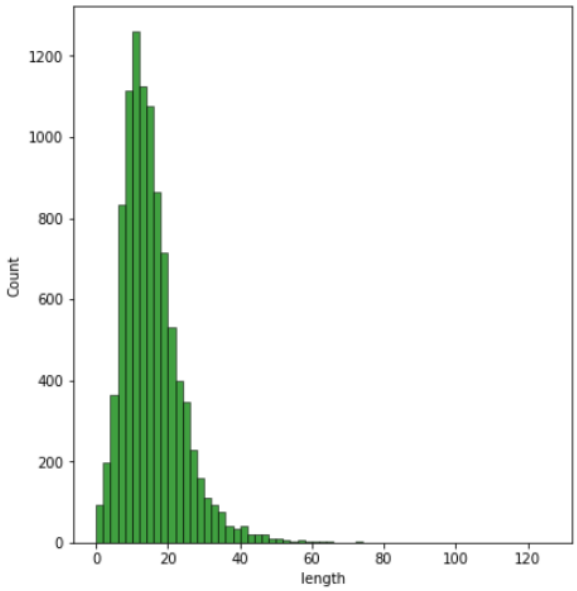


Figure 4.7: Distribution of text length for simple sents

We can see that among the 9829 simple sentences, the average size of each text in terms of the number of words is approximately 15, with a standard deviation of 8.7. The smallest sentence contains

only one word and the longest contains 128 words. The first, second, and third quartiles of sentence lengths are 9, 13, and 19 respectively.

	length
count	12117.0
mean	19.85
std	11.63
min	1.0
25%	12.0
50%	18.0
75%	26.0
max	132.0

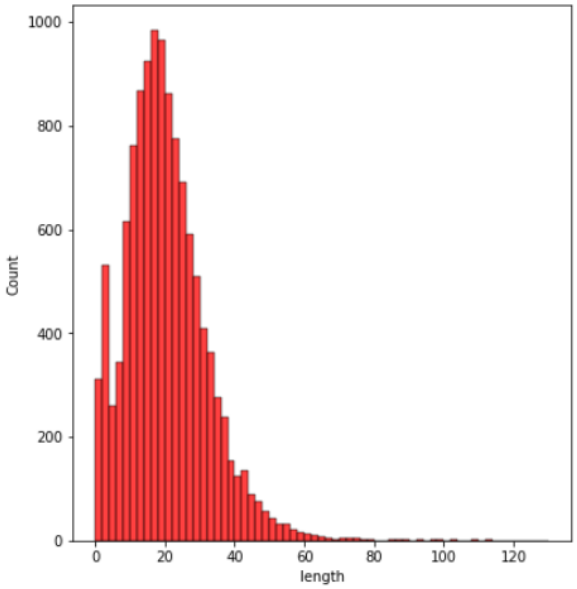


Figure 4.8: Distribution of text length for complex sents

Among the 12117 complex sentences, the average size of each text in terms of the number of words is approximately 20, with a standard deviation of 11.6. The smallest sentence contains only one word and the longest contains 132 words. The first, second, and third quartiles of sentence lengths are 12, 18, and 26 respectively.

Before starting the modeling step, we transform the texts into vectors by simply using the TF-IDF transformations (see Chapter 1), considering only the stems (pieces of words without prefixes). We present below the network used.

The figure below is a neural network architecture called sequential models[6] that combines bidirectional LSTMs and many other layers that have shown good results for sentiment analysis in sentences extracted from tweets. So the first idea was to use similar architecture to hope get similar results.

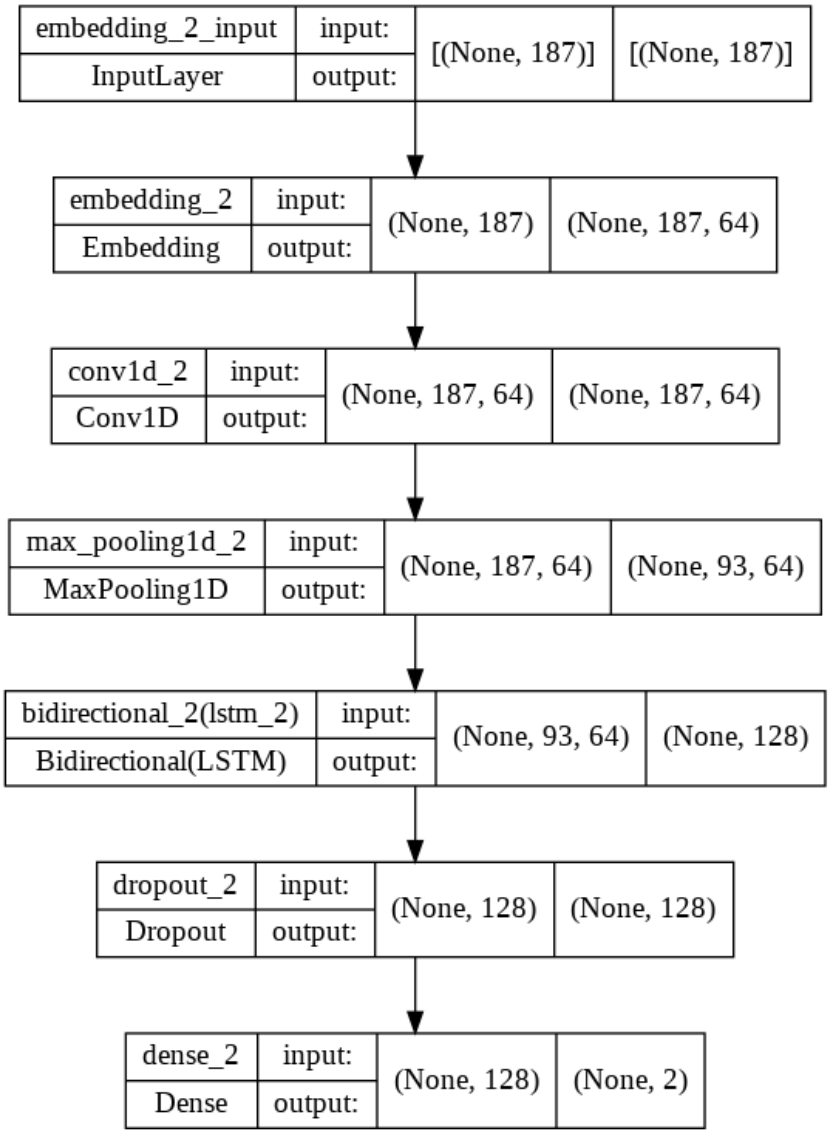
Now the dataset is divided into three parts: A training set to train and estimate the parameters of the model, a validation set to calibrate hyper-parameters, and a test set to estimate the out-of-sample error.

Sample	Data dimensions			
	nb of rows of input	nb of cols input	nb of rows output	nb of classes
Train set (65%)	14923	187	14923	2
Validation Set (20%)	3731	187	3731	2
Test Set (15%)	3292	187	3292	2

Let $n_t = 14923$ denote the total number of observations in the training set. Our objective is to train the network by performing 100 iterations, or epochs, on the entire training dataset using batches of size 128.

[6]The sequential model in TensorFlow enables us to define a neural network in a sequential manner, where the data flows from the input layer to the output layer, passing through a sequence of neural layers. This approach allows us to easily construct deep learning models by stacking multiple layers one after another.

In each epoch, the backpropagation algorithm will be executed and the weights of the network will be updated after processing every 128 data points. This process will be repeated $n_t/128$ times, constituting a single epoch.



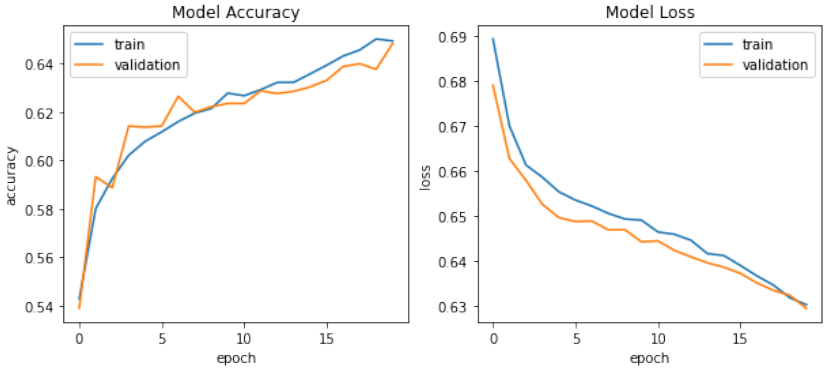
- The input layer has 187 neurons.
- The term none instead of the size of the matrix or tensors used in the iteration refers to a matrix with an unknown number of rows and columns.
- The embedding layer has an input shape of (None, 187) and an output shape of (None, 187, 64). The embedding layer represents each of the 187 features of words as a vector of dimension 64.
- The number of parameters in the embedding layer is determined by the vocabulary size (10000) multiplied by 64. Each word in the vocabulary requires training 64 parameters, resulting in a total of 640000 parameters. The Conv1D layer with a kernel size of 2 and 64 filters takes the output from the embedding layer. The input shape is (None, 187, 64), and the output shape remains (None, 187, 64).

- The Conv1D layer has 8256 parameters, calculated as 64 * (64 * 2 + 1), where the first 64 is the number of output channels, the second 64 is the input channels (embedding vector size), and 2 is the kernel size.
- The MaxPooling1D layer with a pool size of 2 reduces the dimensionality. In this case, the dimension changes from (None, 187, 64) to (None, 93, 64), where the original size is divided by the pool size.
- Each epoch does not train all 14923 data points at once. Instead, the training data is divided into batches of 128 data points, and the model is trained on these batches in random order. An epoch refers to a complete pass through the entire training dataset.

Here is a summary with the number of parameters for each type of layer. We can see that the number of parameters is high, but nowadays with GPUs, this number of 714562 is still reasonable.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, 187, 64)           640000

 conv1d_2 (Conv1D)           (None, 187, 64)           8256

 max_pooling1d_2 (MaxPooling  (None, 93, 64)           0
 1D)

 bidirectional_2 (Bidirectio  (None, 128)              66048
 nal)

 dropout_2 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 2)                 258

=================================================================
Total params: 714,562
Trainable params: 714,562
Non-trainable params: 0
```

The left-hand graph in the figure below represents the accuracy evolution during the training steps (epochs). We can see that it stopped at around 0.64. The right-hand graphic represents the loss variation during training epochs.



The precision obtained when considering all the texts together (without specializing on a specific theme) is **0.6392**. This is a rather low and poor result.
Now the same processing will be done many times until achieving better results, by modifying one or many of the steps of each trial, either the extraction, the embedding, or the neural network architectures. Results will be summarized in Table 4.1.

Theme	Dataset	Embedding	Model	Accuracy
(1) General	$n_s = 9829$ and $n_c = 12117$	Bag of Words	Sequential (NN)	0.6392
	$n_s = 9829$ and $n_c = 12117$	TF-IDF	Sequential (NN)	0.6979
(2) Specific	$n_s = 9911$ and $n_c = 9307$	TF-IDF	Sequential (NN)	0.8499
	$n_s = 9911$ and $n_c = 9307$	TF-IDF	MaxEnt	0.7503
	$n_s = 9911$ and $n_c = 9307$	Features	MaxEnt	0.8616

Table 4.1: **Results** are displayed in the Accuracy column, where:
- n_s (resp. n_c) denotes the number of observed simple texts (resp. complex texts).
- The embedding Bag of Words means we only represent each sentence using a vector of size of the vocabulary with 1 in the positions of the words that the sentence contains and 0 elsewhere.
- The Feature embedding uses similar features to those used for POS-tagging described in 4.2.2.
- The specific theme or field that was used is the following (Movie, Actor, and Authors).

In the above result table 4.1, we have only presented the important outcomes. However, in practice, for each row, we have tested different sets of parameters such as layer width, number of neurons, the depth of the network, the type of embedding, etc. It is important to keep in mind that only significant results corresponding to specific choices (of the parameters of the network) are displayed here.

Surprisingly, regardless of the network used, penalized maximum entropy models on features, consistently outperform neural networks in our context. However, the same method behaves poorly when considering TF-IDF vectorizer; this can be explained by two facts: first the size of the vectors are much smaller in that case. Second TF-IDF does not provide the flexibility to incorporate additional features such as part-of-speech (POS) tag information. The only drawback with Maximum Entropy models is that the features need to be manually engineered and the computation of the "optimal penalty" may be long. Despite this fact, the computation time is better for penalized MaxEnt than for neural networks. Additionally, we suspect that the true potential of neural networks can only be realized when working with a larger amount of data and a significant number of parameters.

Generating simple version

We have also attempted to develop an automatic text simplification tool. For this purpose, we have revisited the extraction step, taking into consideration all the indicators mentioned above, with a particular focus on indicator 3 (which aims to ensure that the texts in both versions of the summary are balanced in terms of the number of words as well as their intersections).

To achieve this, we used 949 pairs of texts in both complex and simple versions. Here is an example of an extracted pair, with x_c representing the complex text and x_s its corresponding simple text:

- x_s = "Zero (0) is a special number. If there are zero things, then there is nothing at all. For example, if a person has zero hats, that means they do not have any hats... In India, zero was theorized in the seventh century by the mathematician Brahmagupta."
- x_c = "0 (zero) : is a number representing an empty quantity. In place-value notation such as the Hindu–Arabic numeral system, 0 also serves as a placeholder numerical digit, which works by multiplying digits to the left of 0 ... and cipher, have also been used."

We finally selected 499 that take into consideration all the indicators mentioned above. After examining the texts, it is evident that still need some more cleaning. Punctuation marks have been

separated by spaces, and the text has been converted to lowercase. This preprocessing step should help save time, but further preprocessing is still required for the text (such as aligning sentences or sequences).

Notice that the complexity of the task is influenced by the complexity of the vocabulary used. A more intricate vocabulary corresponds to a more challenging problem. Now, let's assess the complexity of the dataset we will be dealing with.

- Complex
  - 47698 complex words.
  - 9139 unique complex words.
  - 10 Most common words in the complex dataset: "," "." "the" "of" "in" "and" "a" "was" "is" "(" ")" "to" "by" "The" "as" "on" "an" "for" "from" "with"
- Simple
  - 58304 simple words.
  - 7434 unique simple words.
  - 10 Most common words in the simple dataset: "." "," "the" "of" "in" "and" "was" "a" "is" "(" ")" "He" "to" "The" "by" "It" "on" "an" "for" "at"

More processing is needed, such as completing small sentences using a generic term (here we used the word " < PAD > ") to ensure that all sentences have the same length.

Here is an example of the generated text using a simple basic recurrent neural network after deleting < PAD > :

```
theology grow only of taste;
the most suffering into everything apart
from what responsibility with which a freedom gradations wicked, a personality
of its
own consception, which has altailed, for their own intellectual prejudiced, not as reing the great life faith in sou
th it, if i so
called view o
```

Figure 4.9: Generated example

It is important to note that RNNs do not possess long-term memory capabilities like LSTM networks. This means that RNNs require more information to move away from their current state or central word. In other words, if the network enters an absorbing state, it will continue generating the same output repeatedly. For some examples, the result was a bit more convincing, but it is still weak compared to the latest tools such as chatGPT or BLOOM. In the next steps, one of the potential approaches would be to utilize a larger model that combines different methods. This integration could further enhance the overall performance and effectiveness of the system.

Appendix

Code

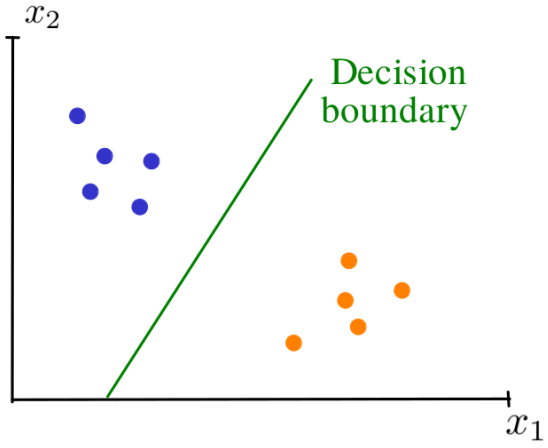
You'll find all the necessary computer material (code and python programs) on my web page[7], **Resources** section - *Programming* tab.

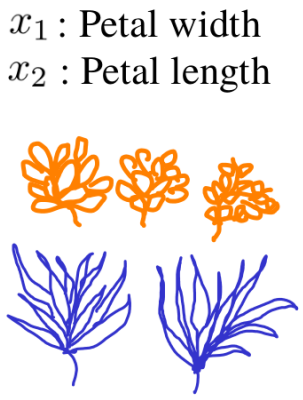
Neural networks

Perceptron (One neuron)

The perceptron is the basic unit of neural networks. It is a binary classification model, capable of linearly separating two classes of points. Let's consider the example where we have two types of plants, toxic plants that we note $y = 1$ and other non-toxic plants that we note $y = 0$. Suppose we are able to measure some attributes of these plants such as the length and width of their leaves (x_1 and x_2 respectively). By representing the measurements in a graph, we observe that the two classes of plants are linearly separable.

y	x_1	x_2
0	0.25	2
0	1.25	2.3
0	0.5	3
0	0.73	2.8
0	0.9	2.1
1	3	0.8
1	2	1
1	3.2	0.4
1	1.9	0.5
1	1.4	0.35





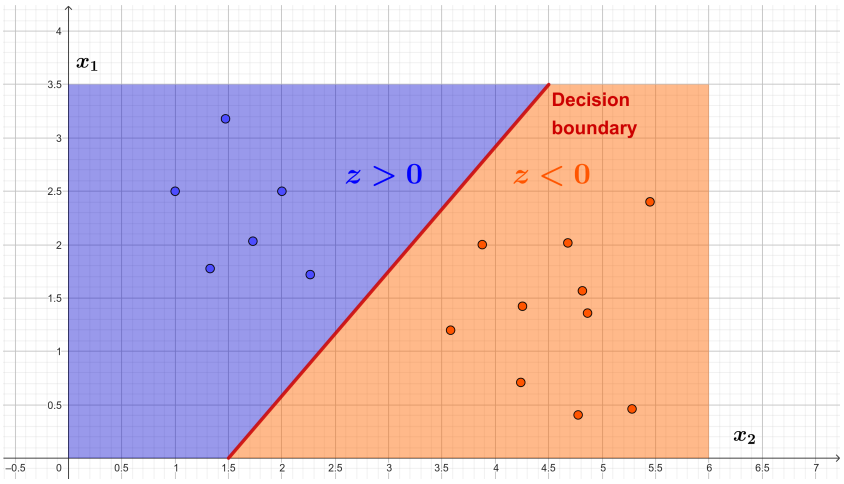
We can therefore develop a model capable of predicting to which class a future plant belongs based on this frontier line, which is called the decision boundary. We can therefore develop a model capable of predicting to which class a future plant belongs based on this line, which is called the decision frontier. If a plant is on the left, it will be considered toxic belonging to the class $y = 1$, and if not, it will be considered non-toxic ($y = 0$). So we will have to find the equation of this line. For that, we will develop what we call a linear model by providing the variables x_1 and x_2 to a neuron, and by multiplying each input of the neuron by a weight w. In this neuron, we will also pass a complementary coefficient called

[7] https://issouani.perso.math.cnrs.fr/

the bias, which gives us a function called $z(x_1, x_2) = w_1x_1 + w_2x_2 + b$.

$$\begin{cases} y_{pred} = 0 & \text{if } z < 0 \\ y_{pred} = 1 & \text{if } z \geq 0 \end{cases}$$

Therefore, on the graph, we can color the regions where this function returns a positive value and the ones where it returns a negative value. We can then see that the decision frontier corresponds to the values of x_1 and x_2 for which $z(x_1, x_2) = 0$. And here we have the equation of our decision frontier. Thus, to predict to which point a future plant belongs, we will have to adjust the parameters w and b in order to separate our two classes as well as possible, then we will be able to say if a plant is in class 0 or 1 by simply looking at the sign of z.



So this is how the perceptron works, the first neuron in the history of deep learning. Now to improve this model, a good thing to do would be to accompany each prediction with a probability. The further a plant is from the decision frontier, the more obvious (probable or likely) it will be that it belongs to its class. One of the functions that allow us to do this is the sigmoid function (also called logistic function), whose expression is $a(z) = (1 + e^{-z})^{-1}$, this function allows us to transform the function (or the signal z) into a probability $a(z)$ that a plant belongs to class 1. For example, if we have a plant for which $z = 1.4$, then this gives a probability $a(z) = 0.8$, which means that according to this model, this plant has an 80% chance of belonging to class 1. This is a relatively high probability, which is logical since this plant is located on the right side of the decision frontier, where we are supposed to have toxic plants. Conversely, if we have a value of z equal to -2.1, then this gives a probability $a(z) = 0.1$ which means that according to our model, this plant has a 10% chance of belonging to class 1, which is consistent.

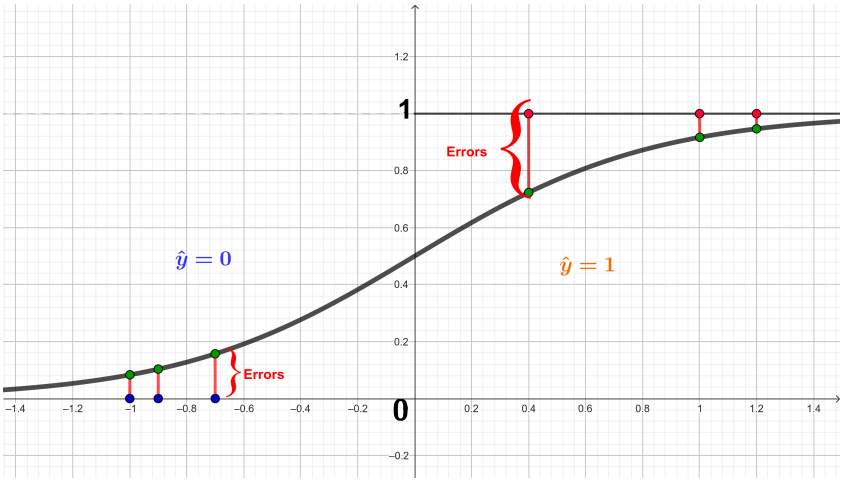
Finally, the binary random variable Y that takes the value 1 (for toxic plants) with probability $a(z)$ is actually a variable that follows the Bernoulli distribution of parameter $a(z)$ with

$$\mathbb{P}[Y = y] = a(z)^y \times (1 - a(z))^{1-y} \quad \text{for } y \in \{0, 1\}.$$

To summarize, inside a neuron we find a linear function $z = w_1x_1 + w_2x_2 + b$ followed by an activation function (the simplest being the sigmoid function) which returns a probability according to a Bernoulli distribution. The goal then is to find the coefficients w and b that give the best possible model, that makes the least errors (or the smallest errors) between the outputs $a(z)$ and the observed data y. And for this, we define a cost function that will allow us to measure these errors. First, the likelihood of the model is written

$$L = \prod_{i=1}^{n} a_i^{y_i} \times (1 - a_i)^{1-y_i}$$

where n represents the number of observations and y_i the observed data number i and a_i is the output or prediction number i. In our case, it will be a function that allows us to measure the distances that we see on the graph below in red



Therefore the cost function we will use is the LogLoss $\mathcal{L}$ that is proportional to the log-likelihood $LL = \log L$. Recall that the likelihood indicates the plausibility of the model with respect to the observed data. As all the values a_i and $1 - a_i$ are between 0 and 1, then their product tends to 0. In practice, when we compute the likelihood on thousands of data, it risks giving us results so close to zero that even the memory of our computer will not be able to store this number. Hence the need to use the log-likelihood.

$$LL = \log L = \sum_{i=1}^{n} \left(y_i \log \left(a_i \right) + \left(1 - y_i \right) \log \left(1 - a_i \right) \right)$$

As maximization algorithms do not really exist, we prefer to minimize the negative version of the criterion

$$\mathcal{L} = -\frac{1}{n} \sum_{i=1}^{n} \left(y_i \log \left(a_i \right) + \left(1 - y_i \right) \log \left(1 - a_i \right) \right)$$

and the factor $1/n$ allows us to normalize the results (by calculating the mean instead of the sum). So minimizing the Log Loss $\mathcal{L}$ is the same as maximizing the likelihood of the model, which allows to minimize the errors of our model. And for this, we use the gradient descent algorithm.

Gradient descent

It is one of the most used learning algorithms, it consists in adjusting the values of the parameters w and b in order to minimize the errors of the model, i.e. to minimize the cost function. To do this, it is necessary to determine how this function $\mathcal{L}$ varies according to the different parameters. This is why we calculate a gradient (a derivative) of the cost function. Recall that in one dimension, the derivative of a function indicates how this function varies. If the derivative is negative, this indicates that the function decreases when w increases and that we will therefore have to increase w if we want to reduce the errors. Conversely, if the derivative is positive, this indicates that we must decrease w if we want to minimize $\mathcal{L}$, i.e. reduce the errors. To do this, we use the following updating formula

$$W_{t+1} = W_t - \alpha \frac{\partial \mathcal{L}}{\partial W_t}$$

where w_t represents the parameter w at time t, and α the positive learning step, and $\frac{\partial L}{\partial W_t}$ the gradient (or the partial derivative) at time t. This allows to increase the weight when the gradient is

negative and to decrease it when the gradient is positive. By repeating this formula in a loop, we are thus able to reach the minimum of the cost function by progressively descending its curve, this is why the term gradient descent has been used.

The only condition for this to work is that the cost function must be convex and that it does not have a local minimum on which the algorithm could fail. Now, we could already program everything on the machine, the only problem is that we need to calculate the gradients first. Instead of directly deriving $\mathcal{L}$, after having replaced the functions a_i by their expressions $a(z_i) = 1/(1+e^{-z_i})$ where we would replace the z_i in their turn by their formula $z_i = w_1 x_1 + w_2 x_2 + b$, we use instead successive derivations in the chain as follows

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial w_1}.$$

By developing each of the three terms on the right, we obtain

$$\frac{\partial \mathcal{L}}{\partial a} = -\frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i}{a_i} - \frac{1-y_i}{1-a_i} \right)$$

$$\frac{\partial a}{\partial z} = \frac{e^{-z}}{\left(1+e^{-z}\right)^2} = a\left(1-a\right)$$

$$\frac{\partial z}{\partial w_1} = x_1,$$

leading to

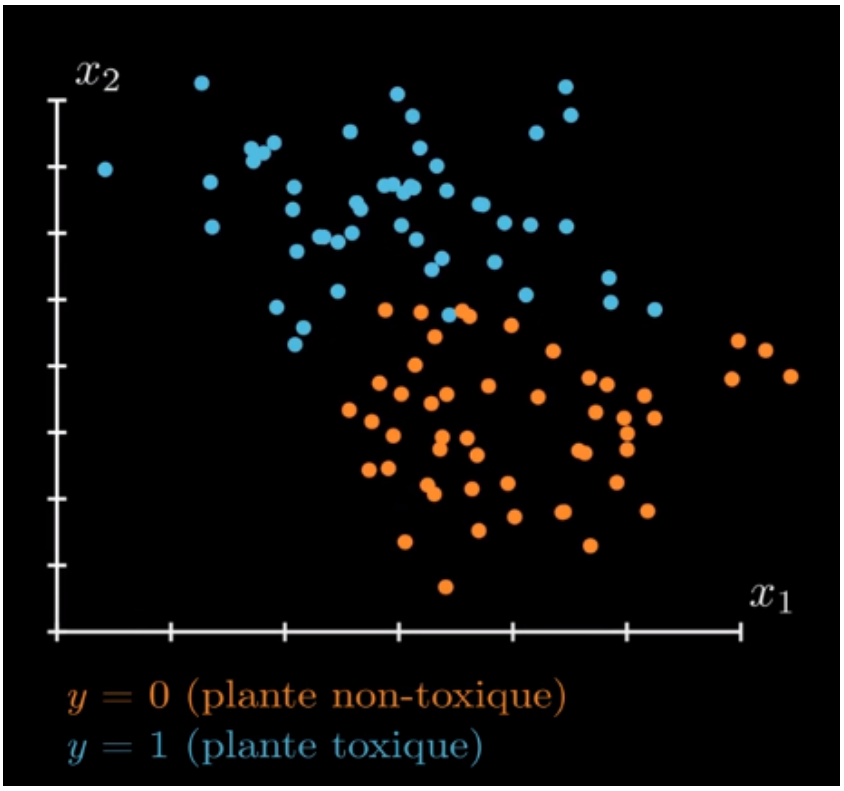
$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w_1} &= -\frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i}{a_i} - \frac{1-y_i}{1-a_i} \right) \times a_i \left(1-a_i\right) \times x_i^{(1)} \\ &= -\frac{1}{n} \sum_{i=1}^{n} \left(y_i \left(1-a_i\right) - \left(1-y_i\right) a_i \right) \times x_i^{(1)} \\ &= -\frac{1}{n} \sum_{i=1}^{n} \left(y_i - a_i \right) x_i^{(1)} \end{aligned}$$

Finally, we get

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w_1} &= -\frac{1}{n} \sum_{i=1}^{n} \left(y_i - a_i \right) x_i^{(1)} \\ \frac{\partial \mathcal{L}}{\partial w_2} &= -\frac{1}{n} \sum_{i=1}^{n} \left(y_i - a_i \right) x_i^{(2)} \\ \frac{\partial \mathcal{L}}{\partial b} &= -\frac{1}{n} \sum_{i=1}^{n} \left(y_i - a_i \right) \end{aligned}$$

Vectorization

Rappelons que l'on dispose d'observations $\{(x_i, y_i)_{i=1\dots n}\}$ avec $x_i = \left(x_i^{(1)}, x_i^{(2)}\right)$



Notations:
Let $V = (V_1, \dots, V_n)$ be a vector of dimention n and M and N two matrices in $\mathcal{M}_n(\mathbb{R})$. The symbols $\sum$, . and $\times$ will denote respectively the sum of elements, classical matrix multiplication, and elementwise multiplication between matrices or vectors

$$\sum V = \sum_{i=1}^{n} V_i = V^T \mathbf{1}_n \text{ where } \mathbf{1}_n = (1, \dots, 1)$$

$$M.N = MN = \{c_{ij}\}_{1 \leq i,j \leq n} \text{ where } c_{ij} = \sum_{k=1}^{n} M_{ik} N_{kj}$$

$$M \times N = \{M_{ij} \times N_{ij}\} \text{ and for vectors } V \times V' = (V_1 V_1', \dots, V_n V_n')$$

With this, we can rewrite all the equations obtained previously as follows

$$\textbf{Model} : \begin{cases} Z = X.W + b \\ A = \frac{1}{1+e^{-Z}} \end{cases}$$

$$\textbf{Costfunction} : \left\{ \mathcal{L} = -\frac{1}{n} \sum_{i=1}^{n} \left(y_i \log(a_i) + (1 - y_i) \log(1 - a_i) \right) \right.$$

$$\textbf{GradientDescent} : \begin{cases} W = W - \alpha \frac{\partial \mathcal{L}}{\partial W} & \left(\frac{\partial \mathcal{L}}{\partial W} = \frac{1}{m} X^T . (A - y) \right) \\ b = b - \alpha \frac{\partial \mathcal{L}}{\partial b} & \left(\frac{\partial \mathcal{L}}{\partial b} = \frac{1}{m} \sum (A - y) \right) \end{cases}$$

This is convenient since it does not only work in dimension 2, but in any dimension. And also because it is convenient to program in machine getting matrix multiplications instead of large for loops especially with huge datasets. The implementation of these functions in the machine, python or other programmation tools becomes very easy.

Multilayers neural network

Similarly, we first have

$$\begin{aligned} Z^{(1)} &= W^{(1)}.X + b^{(1)} \\ A^{(1)} &= \frac{1}{1+e^{-Z^{(1)}}} \\ Z^{(2)} &= W^{(2)}.A^{(1)} + b^{(2)} \\ A^{(2)} &= \frac{1}{1+e^{-Z^{(2)}}} \\ \mathcal{L} &= -\frac{1}{n}\sum\left(y \times \log\left(A^{(2)}\right) + (1-y) \times \log\left(1-A^{(2)}\right)\right) \end{aligned}$$

Therefore

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial W^{(1)}} &= \frac{\partial \mathcal{L}}{\partial A^{(2)}} \times \frac{\partial A^{(2)}}{\partial Z^{(2)}} \times \frac{\partial Z^{(2)}}{\partial A^{(1)}} \times \frac{\partial A^{(1)}}{\partial Z^{(1)}} \times \frac{\partial Z^{(1)}}{\partial W^{(1)}} \\ \frac{\partial \mathcal{L}}{\partial b^{(1)}} &= \frac{\partial \mathcal{L}}{\partial A^{(2)}} \times \frac{\partial A^{(2)}}{\partial Z^{(2)}} \times \frac{\partial Z^{(2)}}{\partial A^{(1)}} \times \frac{\partial A^{(1)}}{\partial Z^{(1)}} \times \frac{\partial Z^{(1)}}{\partial b^{(1)}} \end{aligned}$$

$$dZ2 = \frac{\partial \mathcal{L}}{\partial A^{[2]}} \times \frac{\partial A^{[2]}}{\partial Z^{[2]}} \qquad dZ1 = dZ2 \times \frac{\partial Z^{[2]}}{\partial A^{[1]}} \times \frac{\partial A^{[1]}}{\partial Z^{[1]}}$$

$$\frac{\partial \mathcal{L}}{\partial W^{[2]}} = dZ2 \times \frac{\partial Z^{[2]}}{\partial W^{[2]}} \qquad \frac{\partial \mathcal{L}}{\partial W^{[1]}} = dZ1 \times \frac{\partial Z^{[1]}}{\partial W^{[1]}}$$

$$\frac{\partial \mathcal{L}}{\partial b^{[2]}} = dZ2 \times \frac{\partial Z^{[2]}}{\partial b^{[2]}} \qquad \frac{\partial \mathcal{L}}{\partial b^{[1]}} = dZ1 \times \frac{\partial Z^{[1]}}{\partial b^{[1]}}$$

Bibliography

[1] M. Abramovitch and I. Stegun. Handbook of mathematical tables. *National Bureau of Standards, Washington, DC*, 1970.

[2] J. Alegria. Deafness and reading. In *Handbook of children's literacy*, pages 459–489. Springer, 2004.

[3] T. E. Allen et al. Patterns of academic achievement among hearing impaired students: 1974 and 1983. *Deaf children in America*, 161205, 1986.

[4] S. Aluísio and C. Gasperin. Fostering digital inclusion and accessibility: the porsimples project for simplification of portuguese texts. In *Proceedings of the NAACL HLT 2010 Young Investigators Workshop on Computational Approaches to Languages of the Americas*, pages 46–53, 2010.

[5] A. Arnauld. *Grammaire générale et raisonnée...* Prault fils, 1756.

[6] S. Arora, Y. Liang, and T. Ma. A simple but tough-to-beat baseline for sentence embeddings. In *International conference on learning representations*, 2017.

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

[8] Y. Bar-Hillel. The present status of automatic translation of languages. *Advances in computers*, 1:91–163, 1960.

[9] F. Bartolucci. A penalized version of the empirical likelihood ratio for the population mean. *Statistics & probability letters*, 77(1):104–110, 2007.

[10] J. D. Belder and M.-F. Moens. A dataset for the evaluation of lexical simplification. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 426–437. Springer, 2012.

[11] Y. Bengio, R. Ducharme, and P. Vincent. A neural probabilistic language model. *Advances in neural information processing systems*, 13, 2000.

[12] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71, Mar. 1996.

[13] P. Bertail. Empirical likelihood in some semiparametric models. *Bernoulli*, 12(2):299–331, 2006.

[14] P. Bertail, E. Gautherat, and H. Harari-Kermadec. Exponential bounds for multivariate self-normalized sums. *Electronic Communications in Probability*, 13:628–640, 2008.

[15] P. Bertail, E. Gautherat, and H. Harari-Kermadec. Empirical φ-divergence minimizers for hadamard differentiable functionals. In *Topics in Nonparametric Statistics*, pages 21–32. Springer, 2014.

[16] P. Bertail, E. Gautherat, and H. Harari-Kermadec. Empirical phi-discrepancies and quasi-empirical likelihood: exponential bounds. *ESAIM: Proceedings and Surveys*, 51:212–231, 2015.

[17] P. Bertail, H. Harari-Kermadec, and D. Ravaille. φ-divergence empirique et vraisemblance empirique generalisee. *Annales d'Economie et de Statistique*, pages 131–157, 2007.

[18] D. Biber. Representativeness in Corpus Design. *Literary and Linguistic Computing*, 8(4):243–257, 01 1993.

[19] O. Biran, S. Brody, and N. Elhadad. Putting it simply: a context-aware approach to lexical simplification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 496–501. Association for Computational Linguistics, 2011.

[20] C. Blake, J. Kampov, A. K. Orphanides, D. West, and C. Lown. Unc-ch at duc 2007: Query expansion, lexical simplification and sentence selection strategies for multi-document summarization. In *Proceedings of Document Understanding Conference (DUC) Workshop*, 2007.

[21] A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *Sixth Workshop on Very Large Corpora*, 1998.

[22] J. M. Borwein and A. S. Lewis. Duality relationships for entropy-like minimization problems. *SIAM Journal on Control and Optimization*, 29(2):325–338, 1991.

[23] J. M. Borwein and A. S. Lewis. Partially finite convex programming, part ii: Explicit lattice models. *Mathematical Programming*, 57:49–83, 1992.

[24] S. Bott, L. Rello, B. Drndarević, and H. Saggion. Can spanish be simpler? lexsis: Lexical simplification for spanish. In *Proceedings of COLING 2012*, pages 357–374, 2012.

[25] S. Boucheron, O. Bousquet, and G. Lugosi. Theory of classification: A survey of some recent advances. *ESAIM: probability and statistics*, 9:323–375, 2005.

[26] T. Brants. Tnt: A statistical part-of-speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, ANLC '00, pages 224–231, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[27] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean. Large language models in machine translation. 2007.

[28] E. Brill. Some advances in transformation-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 1)*, AAAI '94, pages 722–727, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence.

[29] A. Brimer and H. Gross. *Wide-span Reading Test Manual: A Test of Reading Comprehension for Children Within the Age-range 7: 0 to 14: 11.* Nelson, 1972.

[30] M. Broniatowski and A. Keziou. Minimization of ϕ-divergences on sets of signed measures. *Studia Scientiarum Mathematicarum Hungarica*, 43(4):403–442, 2006.

[31] M. Broniatowski and A. Keziou. Divergences and duality for estimation and test under moment condition models. *Journal of Statistical Planning and Inference*, 142(9):2554–2573, 2012.

[32] J. Burstein, J. Shore, J. Sabatini, Y.-W. Lee, and M. Ventura. The automated text adaptation tool. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 3–4, 2007.

[33] A. Candido Jr, E. Maziero, C. Gasperin, T. A. Pardo, L. Specia, and S. M. Aluisio. Supporting the adaptation of texts for poor literacy readers: a text simplification editor for brazilian portuguese. In *Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 34–42. Association for Computational Linguistics, 2009.

[34] M. Carasco and J. Florens. Generalization of gmm to a continuum of moments conditions. *Econometrics Theory*, 16:797–834, 2000.

[35] M. Carrasco and R. Kotchoni. Regularized generalized empirical likelihood estimators. *Preprint*, 2017.

[36] J. Carroll, G. Minnen, Y. Canning, S. Devlin, and J. Tait. Practical simplification of english newspaper text to assist aphasic readers. In *Proceedings of the AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, pages 7–10, 1998.

[37] J. Carroll, G. Minnen, D. Pearce, Y. Canning, S. Devlin, and J. Tait. Simplifying text for language-impaired readers. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, 1999.

[38] R. Chandrasekar and B. Srinivas. Automatic induction of rules for text simplification1. *Knowledge-Based Systems*, 10(3):183–190, 1997.

[39] J. Chang, C. Y. Tang, and T. T. Wu. A new scope of penalized empirical likelihood with high-dimensional estimating equations. *The Annals of Statistics*, 46(6B):3185–3216, 2018.

[40] E. Charniak. A maximum-entropy-inspired parser. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*, 2000.

[41] G. Chasapis, R. Liu, and T. Tkocz. Rademacher–gaussian tail comparison for complex coefficients and related problems. *Proceedings of the American Mathematical Society*, 150(03):1339–1349, 2022.

[42] J. Chen, A. M. Variyath, and B. Abraham. Adjusted empirical likelihood and its properties. *Journal of Computational and Graphical Statistics*, 17(2):426–443, 2008.

[43] L. S. Chen, D. Paul, R. L. Prentice, and P. Wang. A regularized hotelling's t 2 test for pathway analysis in proteomic studies. *Journal of the American Statistical Association*, 106(496):1345–1360, 2011.

[44] N. Chomsky. *International Journal of American Linguistics*, 23(3):234–242, 1957.

[45] N. Chomsky. Logical structure in language. *Journal of the American Society for Information Science*, 8(4):284, 1957.

[46] N. Chomsky. [the development of grammar in child language]: Discussion. *Monographs of the Society for Research in Child development*, pages 35–42, 1964.

[47] M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics, 2002.

[48] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.

[49] M. Coltheart, H. Kucera, et al. Kuçera-francis wordlist:[a] frequency count of the brown corpus of present day american english. *Legacy Collection Digital Museum*, 1961.

[50] R. Conrad. The deaf schoolchild london harper & row. 1979.

[51] A. Crépet, H. Harari-Kermadec, and J. Tressou. Using empirical likelihood to combine data: application to food risk assessment. *Biometrics*, 65(1):257–266, 2009.

[52] I. Csiszár. Information geonetry and alternating minimization procedures. *Statistics and decisions*, 1:205–237, 1984.

[53] I. Csiszár. An extended maximum entropy principle and a bayesian justification. *Bayesian statistics*, 2:83–98, 1985.

[54] I. Csiszar. Why least squares and maximum entropy? an axiomatic approach to inference for linear inverse problems. *The annals of statistics*, 19(4):2032–2066, 1991.

[55] I. Csiszár. Maxent, mathematics, and information theory. In *Maximum entropy and Bayesian methods*, pages 35–50. Springer, 1996.

[56] A. Dai, C. Olah, and Q. Le. Document embedding with paragraph vectors. arxiv 2015. *arXiv preprint arXiv:1507.07998*.

[57] J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The annals of mathematical statistics*, pages 1470–1480, 1972.

[58] F. De Saussure. *Cours de linguistique générale*, volume 1. Otto Harrassowitz Verlag, 1989.

[59] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[60] S. Devlin. The use of a psycholinguistic database in the simplification of text for aphasic readers. *Linguistic databases*, 1998.

[61] S. L. Devlin. *Simplifying natural language for aphasic readers.* PhD thesis, University of Sunderland, 1999.

[62] L. Devroye, L. Györfi, and G. Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 2013.

[63] T. DiCiccio, P. Hall, and J. Romano. Empirical likelihood is bartlett-correctable. *the Annals of Statistics*, pages 1053–1061, 1991.

[64] P. Doukhan and J. R. León. Cumulants for stationary mixing random sequences and applications to empirical spectral density. *Probab. Math. Stat*, 10:11–26, 1989.

[65] B. Drndarević and H. Saggion. Towards automatic lexical simplification in spanish: an empirical study. In *Proceedings of the First Workshop on Predicting and Improving Text Readability for target reader populations*, pages 8–16. Association for Computational Linguistics, 2012.

[66] A. Ekbal, R. Haque, and S. Bandyopadhyay. Maximum entropy based bengali part of speech tagging. *A. Gelbukh (Ed.), Advances in Natural Language Processing and Applications, Research in Computing Science (RCS) Journal*, 33:67–78, 2008.

[67] S. C. Emerson and A. B. Owen. Calibration of the empirical likelihood method for a vector mean. *Electronic Journal of Statistics*, 3:1161–1192, 2009.

[68] A. Feldman and J. Hana. *A resource-light approach to morpho-syntactic tagging.* Brill, 2010.

[69] J. R. Firth. *Papers in Linguistics, 1934-1951.* Oxford University Press, London, 1957.

[70] W. N. Francis. A standard corpus of edited present-day american english. *College English*, 26(4):267–273, 1965.

[71] W. N. Francis and H. Kucera. Brown corpus manual. *Letters to the Editor*, 5(2):7, 1979.

[72] T. François and C. Fairon. An ai readability formula for french as a foreign language. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 466–477. Association for Computational Linguistics, 2012.

[73] A. Freitas. Schema-agnostic queries for large-schema databases: A distributional semantics approach, 2015.

[74] A. Freitas, S. Handschuh, and E. Curry. Distributional-relational models: Scalable semantics for databases. In *2015 AAAI Spring Symposium Series*, 2015.

[75] F. Gamboa and E. Gassiat. Bayesian methods and maximum entropy for ill-posed inverse problems. *The Annals of Statistics*, 25(1):328–350, 1997.

[76] C. Gasperin, L. Specia, T. Pereira, and S. Aluísio. Learning when to simplify sentences for natural text simplification. *Proceedings of ENIA*, pages 809–818, 2009.

[77] A. Golan et al. Information and entropy econometrics-volume overview and synthesis. *Journal of Econometrics*, 138(2):379–387, 2007.

[78] A. Golan, G. Judge, and D. Miller. Maximum entropy econometrics, robust estimation with limited. 1996.

[79] A. Golan and E. Maasoumi. Information theoretic and entropy methods: An overview. *Econometric Reviews*, 27(4-6):317–328, 2008.

[80] Y. Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016.

[81] I. J. Good. Maximum entropy for hypothesis formulation, especially for multidimensional contingency tables. *The Annals of Mathematical Statistics*, 34(3):911–934, 1963.

[82] M. Grendar. Empirical maximum entropy methods. In *AIP Conference Proceedings*, volume 872, pages 419–424. American Institute of Physics, 2006.

[83] S. Guiasu and A. Shenitzer. The principle of maximum entropy. *The mathematical intelligencer*, 7(1):42–48, 1985.

[84] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh. *Feature extraction: foundations and applications*, volume 207. Springer, 2008.

[85] F. K. Hammermeister. Reading achievement in deaf adults. *American Annals of the Deaf*, 116(1):25–28, 1971.

[86] B. Harris. Bi-text, a new concept in translation theory. *Language Monthly*, 54(March):8–10, 1988.

[87] M. Harris. Reading comprehension difficulties in deaf children paper presented at the workshop on comprehension disabilities. *Milan, Italy*, 1994.

[88] Z. Harris. *Methods in Structural Linguistics*. University of Chicago Press, Chicago, 1951.

[89] Z. S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[90] H. O. Hartley and J. Rao. A new estimation theory for sample surveys. *Biometrika*, 55(3):547–557, 1968.

[91] G. E. Hinton, A. Krizhevsky, and I. Sutskever. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25(1106-1114):1, 2012.

[92] N. L. Hjort, I. W. McKeague, and I. Van Keilegom. Extending the scope of empirical likelihood. *The Annals of Statistics*, 37(3):1079–1111, 2009.

[93] W. Hoeffding. Probability inequalities for sums of bounded random variables. In *The collected works of Wassily Hoeffding*, pages 409–426. Springer, 1994.

[94] S. K. Jauhar and L. Specia. Uow-shef: Simplex–lexical simplicity ranking based on contextual and psycholinguistic features. In ** SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 477–481, 2012.

[95] E. T. Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.

[96] E. T. Jaynes. Information theory and statistical mechanics. ii. *Physical review*, 108(2):171, 1957.

[97] E. T. Jaynes. On the rationale of maximum-entropy methods. *Proceedings of the IEEE*, 70(9):939–952, 1982.

[98] I. M. Johnstone. On the distribution of the largest eigenvalue in principal components analysis. *The Annals of statistics*, 29(2):295–327, 2001.

[99] S. Jonnalagadda and G. Gonzalez. Biosimplify: an open source sentence simplification engine to improve recall in automatic biomedical information extraction. In *AMIA Annual Symposium Proceedings*, volume 2010, page 351. American Medical Informatics Association, 2010.

[100] L. Kelly. The interaction of syntactic competence and vocabulary during reading by deaf students. *The Journal of Deaf Studies and Deaf Education*, 1(1):75–90, 1996.

[101] T. Kenter, A. Borisov, and M. De Rijke. Siamese cbow: Optimizing word embeddings for sentence representations. *arXiv preprint arXiv:1606.04640*, 2016.

[102] A. Keziou. Dual representation of φ-divergences and applications. *Comptes Rendus Mathematique*, 336(10):857–862, 2003.

[103] R. Kibble. Introduction to natural language processing. *London: University of London*, 2013.

[104] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[105] S. Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[106] S. Lahiri and S. Mukhopadhyay. A penalized empirical likelihood method in high dimensions. *The Annals of Statistics*, 40(5):2511 – 2540, 2012.

[107] S. N. Lahiri and S. Mukhopadhyay. A penalized empirical likelihood method in high dimensions. *The Annals of Statistics*, 40(5):2511–2540, 2012.

[108] W. Lan and W. Xu. Neural network models for paraphrase identification, semantic textual similarity, natural language inference, and question answering. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3890–3902, 2018.

[109] J. H. Lau and T. Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*, 2016.

[110] B. Laurent and P. Massart. Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pages 1302–1338, 2000.

[111] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

[112] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[113] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[114] Y. LeCun et al. Generalization and network design strategies. *Connectionism in perspective*, 19(143-155):18, 1989.

[115] O. Ledoit and M. Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of multivariate analysis*, 88(2):365–411, 2004.

[116] O. Ledoit and M. Wolf. Quadratic shrinkage for large covariance matrices. *Bernoulli*, 28(3):1519–1547, 2022.

[117] A. Lee. *U-statistics: Theory and Practice*. Routledge, 2019.

[118] R. B. Lees. Syntactic structures, 1957.

[119] C. Léonard. Convex conjugates of integral functionals. *Acta Mathematica Hungarica*, 93(4):253–280, 2001.

[120] C. Léonard. Minimization of energy functionals applied to some inverse problems. *Applied mathematics and optimization*, 44(3):273–297, 2001.

[121] C. Léonard. Minimizers of energy functionals. *Acta Mathematica Hungarica*, 93(4):281–325, 2001.

[122] H. Li, A. Aue, D. Paul, J. Peng, and P. Wang. An adaptable generalization of Hotelling's T^2 test in high dimension. *The Annals of Statistics*, 48(3):1815–1847, 2020.

[123] F. Liese and I. Vajda. *Convex statistical distances*, volume 95. Teubner, 1987.

[124] A.-L. Ligozat, C. Grouin, A. Garcia-Fernandez, and D. Bernhard. Approches à base de fréquences pour la simplification lexicale. In *TALN-RECITAL 2013*, volume 1, pages 493–506, 2013.

[125] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[126] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*, 2018.

[127] W. Locke and A. Booth. Mechanical translation. 1955.

[128] S. Mahapatra and J. Sabat. Comprehension difficulties in reading disabled children. *IOSR Journal of Humanities and Social Science*, 21:16–22, 2016.

[129] C. L. Mallows. Information theory and statistics. *Journal of the Royal Statistical Society: Series A (General)*, 122(3):380–381, 1959.

[130] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999.

[131] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, June 1993.

[132] M. Marschark and P. E. Spencer. *The Oxford handbook of deaf studies, language, and education, vol. 2*. Oxford University Press, 2010.

[133] A. Max. Simplification interactive pour la production de textes adaptés aux personnes souffrant de troubles de la compréhension. In *Actes de la 12ème conférence sur le Traitement Automatique des Langues Naturelles. Articles courts*, pages 469–474, 2005.

[134] A. Max. Writing for language-impaired readers. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 567–570. Springer, 2006.

[135] A. McCallum, D. Freitag, and F. C. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Icml*, volume 17, pages 591–598, 2000.

[136] D. McCarthy and R. Navigli. Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the fourth international workshop on semantic evaluations (SemEval-2007)*, pages 48–53, 2007.

[137] D. McCarthy and R. Navigli. The english lexical substitution task. *Language resources and evaluation*, 43(2):139–159, 2009.

[138] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[139] T. McEnery. *Corpus linguistics*. Edinburgh University Press, 2019.

[140] F. Merlevède, J. Najim, and P. Tian. Unbounded largest eigenvalue of large sample covariance matrices: Asymptotics, fluctuations and applications. *Linear Algebra and its Applications*, 577:317–359, 2019.

[141] W. D. Meurers. On the use of electronic corpora for theoretical linguistics: Case studies from the syntax of german. *Lingua*, 115(11):1619–1639, 2005.

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

[143] T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013.

[144] M. Minsky. A framework for representing knowledge. In R. J. Brachman and H. J. Levesque, editors, *Readings in Knowledge Representation*, pages 245–262. Kaufmann, Los Altos, CA, 1985.

[145] J. Mueller and A. Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[146] P. A. Mykland. Dual likelihood. *The Annals of Statistics*, pages 396–421, 1995.

[147] W. K. Newey and R. J. Smith. Higher order properties of gmm and generalized empirical likelihood estimators. *Econometrica*, 72(1):219–255, 2004.

[148] W. K. Newey and R. J. Smith. Higher order properties of gmm and generalized empirical likelihood estimators. *Econometrica*, 72:219–255, 2017.

[149] T. Otsu. Penalized empirical likelihood estimation of semiparametric models. *Journal of Multivariate Analysis*, 98(10):1923–1954, 2007.

[150] A. Owen et al. Empirical likelihood ratio confidence regions. *The Annals of Statistics*, 18(1):90–120, 1990.

[151] A. B. Owen. Empirical likelihood ratio confidence intervals for a single functional. *Biometrika*, 75(2):237–249, 1988.

[152] D. Panchenko. Symmetrization approach to concentration inequalities for empirical processes. *Annals of Probability*, pages 2068–2081, 2003.

[153] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[154] P. V. Paul and D. W. Jackson. *Toward a psychology of deafness: Theoretical and empirical perspectives*. Allyn & Bacon, 1993.

[155] J. Perrez. Introduction to corpus linguistics: Theoretical and methodological basics. 2021.

[156] S. E. Petersen and M. Ostendorf. *Natural Language Processing Tools for Reading Level Assessment and Text Simplication for Bilingual Education*. Citeseer, 2007.

[157] I. Pinelis et al. Extremal probabilistic problems and hotelling's t2 test under a symmetry condition. *The Annals of Statistics*, 22(1):357–368, 1994.

[158] J. Qin and J. Lawless. Empirical likelihood and general estimating equations. *the Annals of Statistics*, 22(1):300–325, 1994.

[159] S. Quigley and C. King. Reading and deafness. san diego, california: College, 1985.

[160] S. Quigley and P. Paul. Language and deafness. college, 1984.

[161] S. P. Quigley et al. The language structure of deaf children. *Volta Review*, 79(2):73–84, 1977.

[162] S. P. Quigley and P. V. Paul. *Language and deafness*. College Hill Books, 1984.

[163] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[164] A. Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. PhD thesis, Philadelphia, PA, USA, 1998. AAI9840230.

[165] A. Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. PhD thesis, Philadelphia, PA, USA, 1998. AAI9840230.

[166] A. Ratnaparkhi et al. A maximum entropy model for part-of-speech tagging. In *Proceedings of the conference on empirical methods in natural language processing*, volume 1, pages 133–142. Philadelphia, PA, 1996.

[167] N. L. Robbins. *The effects of signed text on the reading comprehension of hearing impaired children*. PhD thesis, The University of Nebraska-Lincoln, 1981.

[168] R. Rockafellar. Integrals which are convex functionals. *Pacific journal of mathematics*, 24(3):525–539, 1968.

[169] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[170] R. Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. 1996.

[171] H. Saggion, S. Štajner, S. Bott, S. Mille, L. Rello, and B. Drndarevic. Making it simplext: Implementation and evaluation of a text simplification system for spanish. *ACM Transactions on Accessible Computing (TACCESS)*, 6(4):1–36, 2015.

[172] G. Salton and M. J. McGill. *Introduction to modern information retrieval*. mcgraw-hill, 1983.

[173] R. C. Schank. Conceptual dependency: A theory of natural language understanding. *Cognitive psychology*, 3(4):552–631, 1972.

[174] R. C. Schank and Y. Wilks. The goals of linguistic theory revisited. *Lingua*, 34(4):301–326, 1974.

[175] R. Serfling. Approximation theorems of. *Mathematical Sratistics*, 1980.

[176] S. C. Shapiro. The sneps semantic network processing system. In *Associative networks*, pages 179–203. Elsevier, 1979.

[177] M. Shardlow. A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications*, 4(1):58–70, 2014.

[178] C. M. Shewan and G. J. Canter. Effects of vocabulary, syntax, and sentence length on auditory comprehension in aphasic patients. *Cortex*, 7(3):209–226, 1971.

[179] Z. Shi. Econometric estimation with high-dimensional moment equalities. *Journal of Econometrics*, 195(1):104–119, 2016.

[180] A. Siddharthan. Syntactic simplification and text cohesion. *Research on Language and Computation*, 4(1):77–109, 2006.

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

[182] J. F. Sowa. Conceptual graphs for a data base interface. *IBM Journal of Research and Development*, 20(4):336–357, 1976.

[183] L. Specia, S. K. Jauhar, and R. Mihalcea. Semeval-2012 task 1: English lexical simplification. In ** SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 347–355, 2012.

[184] A. Stefanowitsch. *Corpus linguistics: A guide to the methodology*. Language Science Press, 2020.

[185] R. Swanwick and L. Watson. Literacy in the homes of young deaf children: Common and distinct features of spoken language and sign bilingual environments. *Journal of Early Childhood Literacy*, 5(1):53–78, 2005.

[186] G. Szarvas, C. Biemann, and I. Gurevych. Supervised all-words lexical substitution using delexicalized features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1131–1141, 2013.

[187] M. Talagrand. The missing factor in hoeffding's inequalities. *Annales de l'IHP Probabilités et statistiques*, 31(4):689–702, 1995.

[188] A. Taylor, M. Marcus, and B. Santorini. The penn treebank: an overview. In *Treebanks*, pages 5–22. Springer, 2003.

[189] I. Tellier. Introduction au taln et à l'ingénierie linguistique. *Polycopié de cours: Univer-sité de Lille*, 3, 2010.

[190] I. Tenney, D. Das, and E. Pavlick. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*, 2019.

[191] D. R. Thomas and G. L. Grunkemeier. Confidence interval estimation of survival probabilities for censored data. *Journal of the American Statistical Association*, 70(352):865–871, 1975.

[192] A. N. Tikhonov. On the regularization of ill-posed problems. In *Doklady Akademii Nauk*, volume 153, pages 49–52. Russian Academy of Sciences, 1963.

[193] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 252–259, 2003.

[194] K. Toutanvoa and C. D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora*, pages 63–70, 2000.

[195] N. D. Tracy, J. C. Young, and R. L. Mason. Multivariate control charts for individual observations. *Journal of Quality Technology*, 24(2):88–95, 1992.

[196] A. Turing. Machines informatiques et intelligence. *Mind*, 49:433–460, 1950.

[197] A. M. Turing. Mind. *Mind*, 59(236):433–460, 1950.

[198] J. Van Campenhout and T. Cover. Maximum entropy and conditional probability. *IEEE Transactions on Information Theory*, 27(4):483–489, 1981.

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

[200] J. Véronis. Inf z18-informatique et linguistique i. *Université de Provence. En ligne.¡ http://sites. univ-provence. fr/veronis/cours/INFZ18*, 2001.

[201] D. Vickrey and D. Koller. Sentence simplification for semantic role labeling. In *Proceedings of ACL-08: HLT*, pages 344–352, 2008.

[202] H. Vinod. Maximum entropy measurement error estimates of singular covariance matrices in undersized samples. *Journal of Econometrics*, 20(2):163–174, 1982.

[203] J. von Neumann. Refraction, intersection and reflection of shock waves. *NAVORD Rep. 203-45*, 1945.

[204] J. Von Neumann. First draft of a report on the edvac, 1945. *Reprinted in The Origins of Digital Computers Selected Papers,*, pages 355–364, 1975.

[205] S. S. Wilks. The large-sample distribution of the likelihood ratio for testing composite hypotheses. *The Annals of Mathematical Statistics*, 9(1):60–62, 1938.

[206] G. Wisniewski, L. Denoyer, and P. Gallinari. Restructuration automatique de documents dans les corpus semi-structurés hétérogènes. *Revue des Nouvelles Technologies de l'Information*, Extraction et gestion des connaissances (EGC'2005), Actes des cinquièmes journées Extraction et Gestion des Connaissances, Paris, France, 18-21 janvier 2005, 2 Volumes, RNTI-E-3:227–238, 2005.

[207] G. Wisniewski, L. Denoyer, F. Maes, and P. Gallinari. Modèle probabiliste pour l'extraction de structures dans les documents semistructurés - application aux documents web. In *CORIA*, 2006.

[208] G. Wisniewski, F. Maes, L. Denoyer, and P. Gallinari. Modèle probabiliste pour l'extraction de structures dans les documents web. *Document numérique*, 10(1):89–107, 2007.

[209] K. Wołk and K. Marasek. Building subject-aligned comparable corpora and mining it for truly parallel sentence pairs. *Procedia Technology*, 18:126–132, 2014.

[210] M. Yatskar, B. Pang, C. Danescu-Niculescu-Mizil, and L. Lee. For the sake of simplicity: Unsupervised extraction of lexical simplifications from wikipedia. *arXiv preprint arXiv:1008.1986*, 2010.

[211] D. Yogatama. *Sparse models of natural language text.* PhD thesis, Ph. D. thesis, Carnegie Mellon University, 2015.

[212] T. Young, D. Hazarika, S. Poria, and E. Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[213] F. Yvon. Une petite introduction au traitement automatique des langues naturelles. In *Conference on Knowledge discovery and data mining*, pages 27–36, 2010.

[214] S. Zárate. Subtitling for deaf children on british television. *The Sign Language Translator and Interpreter*, 2(1):15–34, 2008.

[215] S. Zárate. Bridging the gap between deaf studies and avt for deaf children. In *New insights into audiovisual translation and media accessibility*, pages 159–174. Brill, 2010.

[216] Y. Zhang and B. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.

[217] S. Zhao, T. Liu, X. Yuan, S. Li, and Y. Zhang. Automatic acquisition of context-specific lexical paraphrases. In *IJCAI*, volume 178921794, 2007.

[218] Z. Zong and C. Hong. Research on alignment in the construction of parallel corpus. In *Journal of Physics: Conference Series*, volume 1213, page 042003. IOP Publishing, 2019.