

UNIVERSITÀ DEGLI STUDI DI GENOVA



Master's Degree in Applied Mathematics

Master's Thesis

Typo Robustness of Embedding Models: A Metric Stability Analysis

Candidate

Andrea Di Via

Supervisor

Prof. Ulderico Fugacci
IMATI CNR

Company Supervisor

Dott. Claudio Muselli
Rulex S.r.l.

Academic Year 2025/2026

”I meant *no more exams*,
but I typed *new more examples*.^{*}
Persistent curiosity, I guess.”

^{*} *Damerau-Levenshtein distance: 5.*

Abstract

Embedding models are the silent backbone of modern information-retrieval systems: they convert raw text into geometric objects, and the quality of every downstream answer depends on the fidelity of that conversion. Yet the robustness of these models to character-level typographical noise has received surprisingly little systematic attention. This thesis addresses that gap.

I introduce the *Normalised Embedding Displacement* (NED), a dimensionless robustness index defined as the ratio of the output-space displacement induced by a perturbation to its Damerau-Levenshtein edit distance from the original query. Treating an embedding model $E : (\Sigma^*, d_{\text{DL}}) \rightarrow (\mathbb{R}^d, \|\cdot\|)$ as a Lipschitz map between metric spaces, NED estimates the local Lipschitz constant of E at scale ρ ; its bi-Lipschitz refinement yields an empirical stability band $(c(\rho), C(\rho))$ and a scalar distortion index $\kappa := C/c$. The framework is applied to six embedding models (**gemini-embedding-001**, **text-embedding-3-large**, **nomic-embed-text**, **intfloat/multilingual-e5-large**, **BAAI/bge-m3** and **nvidia/nemotron-embed-v1-1b**), evaluated on a 20-query probe set across 13 perturbation budgets and four distance functions (cosine, Euclidean, Manhattan, Chebyshev). Typographical variants are generated by a Markov-chain noise channel estimated from two complementary spelling corpora (Birkbeck and Wikipedia) and parameterised by a relative Damerau-Levenshtein budget ρ ensuring cross-query comparability.

Three main findings emerge. First, mean NED decreases monotonically with ρ for all providers, reflecting metric saturation: the embedding space exhausts its directional response within the first 10 to 20% of corruption budget, with provider-specific differences most pronounced in the moderate regime $\rho \in [0.05, 0.15]$. Second, Gemini and multilingual-e5-large emerge as the most robust providers for complementary reasons: Gemini achieves the lowest distortion ($\kappa \leq 10.4$ under cosine at $\rho = 0.07$), while multilingual-e5-large combines the lowest mean cosine NED (0.031 at $\rho = 0.08$) with the most regular geometric trajectory. Ollama and **bge-m3** exhibit the highest cosine distortion ($\kappa = 261$ and 415 respectively), arising from a sub-cluster topology that causes discontinuous boundary crossings at modest budgets. Third, cosine distance preserves nearest-neighbour ordering under perturbation better than ℓ_p alternatives, despite exhibiting the largest distortion index: angular and norm-based stability are related but non-equivalent criteria.

These findings are connected to end-to-end behaviour through a RAG evaluation on the Rulex Platform documentation. The results reveal a sharp retrieval/generation dissociation: in-domain Jaccard overlap collapses by approximately 78% over the tested range while BERTScore F_1 declines by only 10%, showing that retrieval failures are largely absorbed by the generator and invisible to answer-quality metrics alone. The practical implication is direct: provider selection should be driven by the specific retrieval task, and embedding stability and retrieval stability must be evaluated jointly.

Contents

Introduction	1
1 Mathematical Background	4
1.1 Metric Spaces and Normed Vector Spaces	5
Metric Spaces	5
Normed Spaces and Inner Product Spaces	7
L^p and ℓ_p Spaces	9
Finite-dimensional Normed Spaces	13
Riemannian Manifolds	18
1.2 Edit Distances	19
String Space and Approximate Matching	19
General edit distance properties	20
Main Edit Distances	23
1.3 Markov Chains and Typographical Noise	27
Finite Markov Chains	27
Noise Model	38
Matrix Structure and Spectral Analysis	41
1.4 Summary	45
2 Introduction to AI Systems	46
2.1 Embedding Models	47
The latent space	47
From words to dense vectors	48
Training with contrastive objectives	52
Embedding models as metric maps	53
2.2 The Transformer	55
The Transformer architecture	55

BERT and encoder-only models	60
2.3 Generative Large Language Models	63
Training	65
Limitations	65
2.4 Retrieval-Augmented Generation	66
The RAG pipeline	66
AI agents and agentic RAG	68
Role of the embedding model in RAG	68
2.5 Summary	70
3 Problem Modeling and Experimental Methodology	71
3.1 Problem Statement	71
3.2 Typographical Noise	73
Modeling	73
Dataset and Evaluation Metrics	74
3.3 Embedding Stability	79
Modeling	79
Dataset and Evaluation Metrics	81
3.4 RAG Evaluation	85
Modeling	85
Dataset and Evaluation Metrics	87
3.5 Summary	91
4 Experimental Results	93
4.1 Typographical Noise Characterisation	95
Character Transition Matrix and Transition Graph	95
Ergodicity and Stationary Distribution	97
Confusability Analysis	99
4.2 Embedding Stability Results	101
Input-Output Distance Comparison	102
Displacement Thresholds	109
Bi-Lipschitz Band and Mean NED	111
Distortion Index and Robustness Ranking	115
Geometric Saturation Analysis (t-SNE)	118

4.3	RAG Quality Results	121
	Aggregate Retrieval and Answer Degradation	122
	Distance-Metric Dependence	124
	Provider Dependence	126
4.4	Summary	131
	Conclusion	135
	Appendix	141

Introduction

The way humans interact with machines has changed dramatically in the span of just a few decades. Natural language, once a uniquely human medium, has become a practical and increasingly powerful interface for querying complex systems, retrieving information, and generating knowledge. The landscape changed decisively in 2017, when Vaswani et al. introduced the Transformer architecture [1], a model based entirely on attention mechanisms that proved far more effective at capturing relationships between words than any previous approach. Building on this foundation, BERT [2] established the pre-train and fine-tune paradigm that still dominates the field today, while the GPT series [3] demonstrated that scaling models to billions of parameters could produce systems capable of translating, reasoning, and answering questions without ever being explicitly taught to do so. These systems came to be known as Large Language Models (LLMs), and when ChatGPT was released to the public in November 2022, one hundred million users in two months made it the fastest-adopted application in history. Yet beneath the fluency of these models lies a more fundamental question: *how does the system know what to say?*

In many real-world deployments, LLMs do not rely on their internal knowledge alone. Retrieval-Augmented Generation (RAG) has emerged as a dominant approach for grounding LLM outputs to external, updatable knowledge bases [4]. Rather than relying on what a model learned during the training phase, RAG systems retrieve relevant documents at inference time and use that retrieved content to reduce hallucinations and to produce more accurate and grounded answers. This separation between knowledge storage and the model parameters themselves has made RAG a turning point for question answering, enterprise search, and knowledge-intensive NLP tasks. The quality of the system’s answers, however, is only as good as the quality of what it retrieves.

Before any language model processes a user’s query, an embedding model is responsible for the retrieval step. An embedding model is a neural network that converts text into a vector in a high-dimensional space, making it possible to compare the meaning of different pieces of text through geometric proximity. It is through this step that the user’s intent enters the system: if the retrieved content is incomplete or off-topic, even the most capable LLM will struggle to give a correct response [5]. Embedding models are therefore not just a preprocessing detail; they are a key factor in how well the whole system performs.

Extensive research has focused on improving embedding models for clean, well-written queries [6]. Real-world user inputs, however, are frequently imperfect. Typos, misspellings, and character-level perturbations arise naturally from fast typing, mobile keyboards, and non-native language use. This raises a simple but important

question: *when a user makes a typo, does the embedding model still understand them?*

The answer, as this work demonstrates, is not obvious. While LLM robustness to semantic perturbations has received growing attention [7], the specific sensitivity of embedding models to character-level noise has been studied far less. Magomere et al. [8] have begun to address this gap, showing that standard embedding models suffer significant performance degradation under typos and lexical perturbations, and that strong rerankers cannot fully compensate for failures induced at the retrieval stage. At the sentence encoder level, SenTest [9] further shows that even small character-level perturbations can trigger large, unpredictable shifts in cosine distance, with no consistent relationship between the size of a typo and its effect on the embedding. Additional evidence suggests that architectural choices, such as the type of positional encoding used, also play a role in how sensitive a model is to surface-level changes [10].

Despite all this, a systematic study that evaluates embedding model resilience to typos, across different architectures and across varying levels of noise, has not yet been established. Prior work either looks at the effect of perturbations on downstream task accuracy without examining what happens inside the embedding space [9], or focuses on training new models specifically designed to preserve edit distance [11], without evaluating how modern off-the-shelf, state-of-the-art sentence transformers behave under real-world typos.

This thesis addresses this gap. The main contribution is the definition and evaluation of a robustness index: Normalised Embedding Displacement (NED), that measures how sensitive an embedding model is to character-level noise. The index is computed as the ratio between the distance that a typo causes in the embedding space and the Damerau-Levenshtein edit distance between the original and the perturbed query. A model with a low index handles typos well, keeping the representation of a noisy query close to the original one. A model with a high index, on the other hand, turns small spelling errors into large shifts in the vector space, which can seriously hurt retrieval quality. We evaluate this index across a diverse set of modern embedding models, spanning different architectural families and training paradigms: open-weight encoder models in the BERT tradition, such as `BAAI/bge-m3` and `intfloat/multilingual-e5-large` (both built on XLM-RoBERTa); a locally deployable model with a Mixture-of-Experts feed-forward layer, `nomic-embed-text`; a model distilled from a generative LLaMA decoder into an encoder architecture, `nvidia/llama-nemotron-embed-v1-1b-v2`; and two commercial API-based solutions whose internal architectures are not publicly disclosed: `gemini-embedding-001` (Google) and `text-embedding-3-small` (OpenAI/Azure).

As a secondary contribution, I examine the practical consequences of typos in a RAG setting, comparing BERT-based semantic similarity and Jaccard-based lexical similarity as retrieval methods under increasing levels of noise. Together, these two analyses give both a geometric and a practical view of a problem that affects any system where real users, with all their imperfections, are the ones asking the questions.

Formulating this contribution with any precision required a theoretical apparatus considerably broader than the applied nature of the question might suggest, and it is

worth making explicit which bodies of theory the definition of the index rests upon, since none of them is sufficient on its own. To say that a typographical error is *small* is to presuppose a metric on the space of strings, and to say that the resulting displacement is *proportionally* small is to presuppose the theory of Lipschitz maps between metric spaces and the geometry of the high-dimensional normed spaces in which embeddings reside. To perturb a query as a human being would, rather than at random, is to require a stochastic model of the error channel, and with it the theory of finite Markov chains. And none of this is of any use in isolation from a working understanding of the systems being described, since a robustness index cannot be defined without regard to the metric under which a model will be queried, to the way a retrieval pipeline indexes its corpus, or to the behaviour of the generative model downstream. The mathematical and the computational sides of this work proved inseparable, and the two chapters that follow develop them in turn. The thesis is structured as follows:

- **Chapter 1** builds the mathematical vocabulary in which the central question can be posed at all. The territory it must cover is wide, ranging from the axioms of metric spaces and the geometry of finite-dimensional normed spaces; through the combinatorial theory of edit distances to the theory of finite Markov chains and the Perron-Frobenius argument that validates the confusion matrix as a noise source rather than a mere table of numbers.
- **Chapter 2** provides an introduction to the AI systems involved in this study, namely Embedding Models, Large Language Models (LLMs), and the Retrieval-Augmented Generation paradigm, with particular attention to the geometry of the latent space and to the role of the distance function in retrieval.
- **Chapter 3** describes how the problem is modelled, presenting the datasets used, the evaluation metrics adopted, and the methodology followed to inject and measure noise across different embedding models.
- **Chapter 4** presents the experimental results, analysing how character-level noise propagates through the embedding space and affects retrieval performance across different architectures and noise levels.
- **Conclusion** summarises the main findings and suggests possible directions for future research.

1

Mathematical Background

This chapter collects the mathematical foundations required to formalise the central object of study in this thesis. The main question addressed is the following: *how sensitive is an embedding model E to small perturbations of its input?* Formally, an embedding model is treated as a function between two metric spaces,

$$E : (\Sigma^*, d) \longrightarrow (\mathbb{R}^d, \|\cdot\|),$$

where Σ^* is a space of strings equipped with an edit distance d , and $\mathbb{R}^d$ is a finite-dimensional normed vector space. The chapter is organised as follows:

- **Section 1.1** develops the theory of metric spaces and normed vector spaces: the axioms of a metric, the structure of normed and inner product spaces, L^p spaces and ℓ_p norms, finite-dimensional normed spaces, and an introduction to Riemannian manifolds [12, 13, 14]. The cosine distance, which governs nearest-neighbour retrieval in the embedding space, is also introduced here.
- **Section 1.2** studies edit distances on the space of strings Σ^* : the general properties shared by all edit distances [15], the principal instances used in approximate string matching (Hamming, LCS, Levenshtein, and Optimal String Alignment), and the Damerau-Levenshtein distance [16, 17], which serves as the primary input metric of this thesis and whose metric axioms are verified in detail.
- **Section 1.3** recalls the basic theory of finite Markov chains and stochastic transition matrices, including their spectral structure [18, 19, 20]. This formalism underlies the character-level noise channel employed in Chapter 3 to produce controlled typographical perturbations of query strings.

1.1 Metric Spaces and Normed Vector Spaces

In this thesis, the notion of distance arises at two distinct levels: on the input side, strings are compared via combinatorial edit distances, while on the output side, embedding vectors are compared via norms in $\mathbb{R}^d$. Both settings can be formalised as metric spaces, and embedding models can be interpreted as functions between them. The purpose of this section is to recall the mathematical tools needed to study such functions.

It is worth clarifying, from the outset, the terminology adopted throughout this thesis. I use the term *distance* to denote, in a broad and informal sense, any function that quantifies dissimilarity between two objects. The term *metric* is reserved for the strict notion of Definition 1.1.1, satisfying non-degeneracy, symmetry, and the triangle inequality. Among the four output-space distances used throughout this thesis, three (Euclidean, Manhattan, Chebyshev) are genuine metrics, being induced by norms (Example 1.1.2); the fourth, cosine distance, is a metric only when restricted to the unit sphere $\mathbb{S}^{d-1}$ (Definition 1.1.16), and is not guaranteed to be one on the raw, non-normalised embeddings used in the experimental pipeline (Remark 1.1.6). For this reason, unless the metric property has been explicitly established, I refer to this family collectively as *distance functions* rather than *metrics*.

Metric Spaces

The notion of *distance* plays a central role in many areas of mathematics, providing a quantitative way to compare objects and to analyse the effect of perturbations. Metric spaces offer a general and flexible framework in which this notion can be defined abstractly, independently of any specific structure such as coordinates or linearity.

Within this framework, fundamental analytical concepts such as convergence, continuity, and stability can be developed in a unified way. In particular, metric spaces provide the natural setting for studying how small changes in the input of a function affect its output, a question that lies at the core of modern analysis. I begin with the most basic building block.

Definition 1.1.1 (Metric space). *Let X be a non-empty set. A function $d : X \times X \rightarrow \mathbb{R}$ is called a metric on X if, for all $x, y, z \in X$, the following properties hold:*

- (a) **Non-degeneracy:** $d(x, y) \geq 0$, with equality if and only if $x = y$.
- (b) **Symmetry:** $d(x, y) = d(y, x)$.
- (c) **Triangle inequality:** $d(x, z) \leq d(x, y) + d(y, z)$.

The pair (X, d) is called a metric space.

Remark 1.1.1. *The non-degeneracy axiom combines two requirements: positivity ($d(x, y) \geq 0$) and the identity of indiscernibles ($d(x, y) = 0 \iff x = y$). Together, they ensure that the metric takes strictly positive values between distinct points and vanishes only on the diagonal.*

The metric induces a natural topology on X , which allows us to make precise what it means for a sequence to converge and for a function to vary continuously. These notions will be essential when I discuss the stability of embedding functions.

Definition 1.1.2 (Open ball, open set, and metric topology). *Let (X, d) be a metric space, $x_0 \in X$, and $r > 0$. The open ball of radius r centred at x_0 is*

$$B(x_0, r) := \{x \in X : d(x, x_0) < r\}.$$

A set $U \subseteq X$ is open if for every $x \in U$ there exists $r > 0$ such that $B(x, r) \subseteq U$. The family of open sets induced by the metric defines a topology on X , called the metric topology. This topological structure allows one to define convergence and regularity properties of functions between metric spaces.

Definition 1.1.3 (Continuity). *Let (X, d_X) and (Y, d_Y) be metric spaces. A function $f : X \rightarrow Y$ is continuous at $x \in X$ if for every $\varepsilon > 0$ there exists $\delta > 0$ such that*

$$d_X(x, y) < \delta \implies d_Y(f(x), f(y)) < \varepsilon.$$

It is continuous if it is continuous at every point $x \in X$.

Continuity tells us that nearby inputs produce nearby outputs, but gives no quantitative bound on *how* nearby. To obtain such control, which is precisely what is needed to bound the displacement of an embedding vector under a small edit to the input string, a stronger notion is required.

Definition 1.1.4 (Lipschitz continuity and bi-Lipschitz functions). *A function $f : (X, d_X) \rightarrow (Y, d_Y)$ is Lipschitz continuous if there exists a real constant $L \geq 0$ such that*

$$d_Y(f(x), f(y)) \leq L d_X(x, y) \quad \forall x, y \in X.$$

The infimum of all such constants is called the Lipschitz constant of f , denoted $\text{Lip}(f)$.

A function is bi-Lipschitz if there exist real constants $c, C > 0$ such that

$$c d_X(x, y) \leq d_Y(f(x), f(y)) \leq C d_X(x, y) \quad \forall x, y \in X.$$

Lipschitz continuity provides a global stability bound: small perturbations in the input produce controlled perturbations in the output. The bi-Lipschitz condition adds a lower bound, requiring that distinct points remain quantitatively separated under the function. Thus, while the upper bound controls *stability*, the lower bound controls *discriminability*. This pair of properties is precisely what I will seek to estimate empirically for embedding models in Chapters 3 and 4.

Definition 1.1.5 (Isometry). *A function $f : (X, d_X) \rightarrow (Y, d_Y)$ is an isometry if*

$$d_Y(f(x), f(y)) = d_X(x, y) \quad \forall x, y \in X.$$

Isometries preserve distances exactly, and are therefore bi-Lipschitz with $c = C = 1$. The hierarchy *isometry* $\subset$ *bi-Lipschitz* $\subset$ *Lipschitz continuous* provides a natural scale to measure how much a transformation deforms the geometry of a space. An ideal embedding model would be an isometry; in practice, it will be somewhere between Lipschitz continuous and bi-Lipschitz, and the ratio $\kappa = C/c$ (the *distortion*) measures how far it falls short.

Before studying the geometry of the embedding output space, I need one more property of metric spaces: *completeness*. This ensures that sequences of progressively perturbed inputs have well-defined limiting behaviour, providing a stable framework for the convergence and robustness arguments used later.

Definition 1.1.6 (Cauchy sequence and completeness). *A sequence (x_n) in a metric space (X, d) is a Cauchy sequence if for every $\varepsilon > 0$ there exists $N \in \mathbb{N}$ such that $d(x_m, x_n) < \varepsilon$ for all $m, n > N$. The space (X, d) is complete if every Cauchy sequence converges to a point in X .*

Completeness is a fundamental property in analysis: it guarantees that every Cauchy sequence converges within the space itself, thereby ensuring that operations such as taking limits or solving equations can be carried out without exiting the ambient space. Both spaces appearing in this thesis are complete: the output space $(\mathbb{R}^d, \|\cdot\|)$ by virtue of its finite-dimensional structure (Theorem 1.1.5), and the input space (Σ^*, d) because edit distances are integer-valued, so any Cauchy sequence must eventually be constant (Proposition 1.2.1).

A particularly clean example of a complete metric space is given by discrete metric spaces, whose simple structure will reappear when I will verify completeness of the string space.

Definition 1.1.7 (Discrete metric space). *Let X be a non-empty set. The discrete metric on X is defined by*

$$d(x, y) := \begin{cases} 0 & \text{if } x = y, \\ 1 & \text{if } x \neq y. \end{cases}$$

The pair (X, d) is called a discrete metric space.

Proposition 1.1.1 (Completeness of discrete metric spaces). *Every discrete metric space is complete. More generally, every metric space (X, d) in which d takes values in $\mathbb{N}_{\geq 0}$ is complete.*

Proof. Let (x_n) be a Cauchy sequence in (X, d) . For $\varepsilon = \frac{1}{2}$ there exists K such that $d(x_m, x_k) < \frac{1}{2}$ for all $m, k > K$. Since d is integer-valued, this forces $d(x_m, x_k) = 0$, hence $x_m = x_k$. The sequence is eventually constant and therefore convergent. $\square$

Normed Spaces and Inner Product Spaces

Metric spaces provide a general notion of distance, but the output space of an embedding model carries much more structure: it is a real vector space, and its distance

arises from a *norm*. The norm axioms (non-degeneracy, absolute homogeneity, and the triangle inequality) endow the space with an algebraic notion of length and direction. This additional structure bridges the abstract geometric framework introduced in this section with the concrete computations performed by embedding models. Operations such as cosine similarity, nearest-neighbour search, and robustness estimates under perturbations all fundamentally rely on norm-induced geometry.

Definition 1.1.8 (Normed space). *Let V be a real vector space. A function $\|\cdot\| : V \rightarrow \mathbb{R}$ is a norm on V if, for all $u, v \in V$ and $\lambda \in \mathbb{R}$, the following properties hold:*

- (a) **Non-degeneracy:** $\|v\| \geq 0$, with equality if and only if $v = 0$;
- (b) **Absolute homogeneity:** $\|\lambda v\| = |\lambda| \|v\|$;
- (c) **Triangle inequality:** $\|u + v\| \leq \|u\| + \|v\|$.

The pair $(V, \|\cdot\|)$ is called a normed space.

Every norm induces a metric via $d(u, v) := \|u - v\|$, making every normed space a metric space. The converse is false in general: not every metric arises from a norm. The following lemma identifies the two properties that characterise metrics induced by norms.

Lemma 1.1.1 (Translation invariance). *Let $(V, \|\cdot\|)$ be a normed space and d the metric induced by the norm. Then for all $x, y, a \in V$ and every scalar λ :*

- (a) $d(x + a, y + a) = d(x, y)$;
- (b) $d(\lambda x, \lambda y) = |\lambda| d(x, y)$.

Proof. Both identities follow directly from $d(u, v) := \|u - v\|$:

$$\begin{aligned} d(x + a, y + a) &= \|(x + a) - (y + a)\| = \|x - y\| = d(x, y), \\ d(\lambda x, \lambda y) &= \|\lambda x - \lambda y\| = |\lambda| \|x - y\| = |\lambda| d(x, y). \end{aligned}$$

□

A metric on a vector space arises from a norm only if both conditions of Lemma 1.1.1 hold. The following example shows that this is a genuine restriction.

Example 1.1.1. *The discrete metric on $\mathbb{R}^n$ (Definition 1.1.7) is translation-invariant (a) but fails condition (b):*

$$d(0, 2v) = 1 \neq 2 = 2d(0, v) \quad \forall v \neq 0.$$

Hence it cannot arise from a norm.

Among normed spaces, those that additionally carry an inner product allow one to measure angles between vectors, not just distances. This angular measurement is the geometric operation underlying cosine similarity, which will emerge as the primary retrieval metric once I specialise to the high-dimensional output space $\mathbb{R}^d$.

Definition 1.1.9 (Inner product). *An inner product on a real vector space V is a map $\langle \cdot, \cdot \rangle : V \times V \rightarrow \mathbb{R}$ satisfying, for all $u, v, w \in V$ and $\lambda \in \mathbb{R}$:*

- (a) **Bilinearity:** $\langle \lambda u + v, w \rangle = \lambda \langle u, w \rangle + \langle v, w \rangle$, and similarly in the second argument;
- (b) **Symmetry:** $\langle u, v \rangle = \langle v, u \rangle$;
- (c) **Positive definiteness:** $\langle v, v \rangle \geq 0$, with equality if and only if $v = 0$.

Every inner product induces a norm by setting $\|v\| := \sqrt{\langle v, v \rangle}$; this is well-posed by positive definiteness. A normed space whose norm arises from an inner product is called an *inner product space* (or *pre-Hilbert space*).

The completeness properties of these spaces are summarised by two classical names.

Definition 1.1.10 (Banach and Hilbert spaces). *A complete normed space is called a Banach space [12]. A complete inner product space is called a Hilbert space.*

Every Hilbert space is in particular a Banach space. As shown in Corollary 1.1.1, the converse does not hold: for $p \neq 2$ the space L^p is a Banach space but does not admit an inner product compatible with $\|\cdot\|_p$.

L^p and ℓ_p Spaces

I now make the output space of an embedding model more explicit. Embedding vectors are elements of $\mathbb{R}^d$, and their similarity or dissimilarity can be quantified through different distance functions, including the Euclidean, Manhattan, Chebyshev, and cosine distances. These distances arise from different underlying norms, most of which belong to the broader ℓ_p family. I therefore introduce ℓ_p spaces in their general form before restricting to the finite-dimensional setting relevant for embeddings. The general framework is also needed to state the Fischer-Riesz theorem (Theorem 1.1.3), which guarantees completeness. The definitions and results presented follow [21, 12, 13].

Definition 1.1.11 ($\mathcal{L}^p$ space). *Let $(X, \mathcal{M}, \mu)$ be a measure space and $p \in [1, +\infty]$. The $\mathcal{L}^p$ space is*

$$\mathcal{L}^p(X, \mu) := \left\{ f : X \rightarrow \mathbb{R} \text{ measurable} \mid \|f\|_p < \infty \right\},$$

equipped with

$$\|f\|_p := \begin{cases} \left(\int_X |f|^p d\mu \right)^{1/p} & p < +\infty, \\ \text{ess sup}_{x \in X} |f(x)| & p = +\infty. \end{cases}$$

Definition 1.1.12 (L^p space). *The L^p space is the quotient*

$$L^p(X, \mu) := \mathcal{L}^p(X, \mu) / \sim,$$

where $f \sim g$ if and only if $f = g$ μ -almost everywhere.

The passage to the quotient is necessary because two distinct measurable functions may agree almost everywhere, making $\|\cdot\|_p$ only a semi-norm on $\mathcal{L}^p(X, \mu)$; identifying such functions makes it a genuine norm [13]. This subtlety disappears when the underlying measure is the counting measure.

Definition 1.1.13 (ℓ_p spaces and ℓ_p^n norms). *Let ν be the counting measure on $\mathbb{N}$, defined by $\nu(A) := |A|$. The ℓ_p space is*

$$\ell_p := L^p(\mathbb{N}, \nu) = \left\{ (x_k)_{k \geq 1} \subset \mathbb{R} \mid \|(x_k)\|_p < \infty \right\}.$$

Its finite-dimensional restriction, obtained by taking $X = \{1, \dots, n\}$ with counting measure ν_n , is

$$\ell_p^n := L^p(\{1, \dots, n\}, \nu_n) \cong (\mathbb{R}^n, \|\cdot\|_p),$$

where for $v = (v_1, \dots, v_n) \in \mathbb{R}^n$:

$$\|v\|_p := \begin{cases} \left(\sum_{i=1}^n |v_i|^p \right)^{1/p} & p \in [1, +\infty), \\ \max_{1 \leq i \leq n} |v_i| & p = +\infty. \end{cases}$$

Remark 1.1.2. *Under the counting measure on a finite set, every singleton has positive measure, so no index can be neglected: two functions agreeing ν_n -almost everywhere are pointwise identical, and no quotient is needed. This makes precise that the codomain of an embedding model is always the same vector space $\mathbb{R}^n$, regardless of p ; the ℓ_p structure enters only through the choice of metric used to compare outputs.*

To verify that $\|\cdot\|_p$ is indeed a norm, I need two classical inequalities.

Lemma 1.1.2 (Jensen's inequality). *Let $\varphi : \mathbb{R} \rightarrow \mathbb{R}$ be a convex function, $(\Omega, \mathcal{F}, \mathbb{P})$ a probability space, and $Z : \Omega \rightarrow \mathbb{R}$ an integrable random variable. Then*

$$\varphi(\mathbb{E}[Z]) \leq \mathbb{E}[\varphi(Z)].$$

Proof. Since φ is convex, for every $x_0 \in \mathbb{R}$ there exists a supporting hyperplane: a constant $a \in \mathbb{R}$ such that

$$\varphi(x) \geq \varphi(x_0) + a(x - x_0) \quad \forall x \in \mathbb{R}.$$

Setting $x_0 := \mathbb{E}[Z]$ and $x := Z(\omega)$, integrating both sides with respect to $\mathbb{P}$, and using linearity of expectation gives

$$\mathbb{E}[\varphi(Z)] \geq \varphi(\mathbb{E}[Z]) + a(\mathbb{E}[Z] - \mathbb{E}[Z]) = \varphi(\mathbb{E}[Z]).$$

□

Lemma 1.1.3 (Young's inequality). *Let $p, q \in (1, \infty)$ with $\frac{1}{p} + \frac{1}{q} = 1$, and let $a, b \geq 0$. Then*

$$ab \leq \frac{a^p}{p} + \frac{b^q}{q}.$$

Proof. The function $t \mapsto e^t$ is convex. Applying Jensen's inequality to the two-point distribution with weights $1/p$ and $1/q$ gives

$$ab = e^{\frac{\log a^p}{p} + \frac{\log b^q}{q}} \leq \frac{a^p}{p} + \frac{b^q}{q}.$$

□

Young's inequality is the key tool for proving Hölder's inequality, which in turn is what makes the triangle inequality for $\|\cdot\|_p$ (Minkowski's inequality) work. This chain of implications is worth keeping in mind: it is the reason the ℓ_p norm is a norm for every $p \geq 1$, but not for $p < 1$. The causes of this failure are made explicit after Minkowski's inequality below in Remark 1.1.3.

Theorem 1.1.1 (Hölder's inequality). *Let $p, q \in [1, +\infty]$ with $\frac{1}{p} + \frac{1}{q} = 1$. Then for all $u, v \in \mathbb{R}^n$,*

$$\sum_{i=1}^n |u_i v_i| \leq \|u\|_p \|v\|_q.$$

The case $p = q = 2$ is the Cauchy-Schwarz inequality.

Proof. The proof proceeds by cases (the case $p = \infty, q = 1$ follows by symmetry from the first).

- **Case $p = 1, q = \infty$:**

$$\sum_{i=1}^n |u_i v_i| \leq \|v\|_\infty \sum_{i=1}^n |u_i| = \|u\|_1 \|v\|_\infty;$$

- **Case $p, q \in (1, \infty)$:** if $\|u\|_p = 0$ or $\|v\|_q = 0$ the inequality is trivial. Otherwise, set $\tilde{u}_i := |u_i|/\|u\|_p$ and $\tilde{v}_i := |v_i|/\|v\|_q$, so that $\|\tilde{u}\|_p = \|\tilde{v}\|_q = 1$. Applying Young's inequality (Lemma 1.1.3) termwise and summing over $i = 1, \dots, n$:

$$\sum_{i=1}^n \tilde{u}_i \tilde{v}_i \leq \sum_{i=1}^n \left(\frac{\tilde{u}_i^p}{p} + \frac{\tilde{v}_i^q}{q} \right) = \frac{\|\tilde{u}\|_p^p}{p} + \frac{\|\tilde{v}\|_q^q}{q} = \frac{1}{p} + \frac{1}{q} = 1.$$

Rescaling by $\|u\|_p \|v\|_q$ gives the claim.

□

Theorem 1.1.2 (Minkowski's inequality). *For all $u, v \in \mathbb{R}^n$ and $p \in [1, +\infty]$,*

$$\|u + v\|_p \leq \|u\|_p + \|v\|_p.$$

Proof. The proof proceeds by cases:

- **Case $p = 1$:** $\|u + v\|_1 = \sum_{i=1}^n |u_i + v_i| \leq \sum_{i=1}^n (|u_i| + |v_i|) = \|u\|_1 + \|v\|_1;$

- **Case $p = \infty$:** for every i , $|u_i + v_i| \leq |u_i| + |v_i| \leq \|u\|_\infty + \|v\|_\infty$, so taking the maximum over i gives $\|u + v\|_\infty \leq \|u\|_\infty + \|v\|_\infty$;
- **Case $p \in (1, \infty)$:** if $\|u + v\|_p = 0$ the inequality is trivial. Otherwise, write

$$\|u + v\|_p^p = \sum_{i=1}^n |u_i + v_i|^p \leq \sum_{i=1}^n |u_i + v_i|^{p-1} (|u_i| + |v_i|).$$

Applying Hölder's inequality (Theorem 1.1.1) with exponents p and $q = p/(p-1)$ to each of the two sums separately:

$$\|u + v\|_p^p \leq (\|u\|_p + \|v\|_p) \|u + v\|_p^{p-1}.$$

Dividing both sides by $\|u + v\|_p^{p-1}$ gives the claim. □

Remark 1.1.3. *The proof of the case $p \in (1, \infty)$ applies Hölder's inequality with exponent $q = p/(p-1)$, which requires $p > 1$. For $p < 1$ the exponent q is non-positive, Hölder fails, and the triangle inequality breaks down. The following counterexample makes this concrete: with $p = \frac{1}{2}$, $u = (1, 0)$, and $v = (0, 1)$ in $\mathbb{R}^2$,*

$$\|u + v\|_{1/2} = (1^{1/2} + 1^{1/2})^2 = 4 > 2 = \|u\|_{1/2} + \|v\|_{1/2}.$$

Thus $\|\cdot\|_p$ is not a norm for $p < 1$, confirming that $p \geq 1$ is not merely a technical convenience but a necessary condition.

Proposition 1.1.2 (ℓ_p norms are norms). *For every $p \in [1, +\infty]$, $\|\cdot\|_p$ is a norm on $\mathbb{R}^n$.*

Proof. The three norm axioms are verified for all $p \in [1, +\infty]$:

- Non-degeneracy.** $\|v\|_p \geq 0$ since it is a sum or supremum of non-negative terms. If $v \neq 0$ then $|v_i| > 0$ for some i , so $\|v\|_p \geq |v_i| > 0$. Conversely, $\|0\|_p = 0$.
- Absolute homogeneity.** For $p < \infty$: $\|\lambda v\|_p = (\sum_{i=1}^n |\lambda v_i|^p)^{1/p} = |\lambda| \|v\|_p$. For $p = \infty$: $\|\lambda v\|_\infty = \max_i |\lambda v_i| = |\lambda| \|v\|_\infty$.
- Triangle inequality.** This is Minkowski's inequality (Theorem 1.1.2). □

The three most commonly used instances of the ℓ_p^n norm are collected in the following example for reference; the cosine distance, which occupies a structurally distinct role among the four metrics evaluated in this thesis, is introduced in Definition 1.1.16 once the Hilbert structure of $(\mathbb{R}^n, \|\cdot\|_2)$ has been established.

Example 1.1.2 (ℓ_p distances on $\mathbb{R}^n$). *The most used instances of the ℓ_p norm are:*

- ℓ_2 (Euclidean): $\|v\|_2 = (\sum_{i=1}^n v_i^2)^{1/2}$, induced by the standard inner product;

- ℓ_1 (Manhattan): $\|v\|_1 = \sum_{i=1}^n |v_i|$;
- ℓ_∞ (Chebyshev): $\|v\|_\infty = \max_{1 \leq i \leq n} |v_i|$.

These norms induce different geometries on $\mathbb{R}^n$ but are topologically equivalent (Theorem 1.1.4).

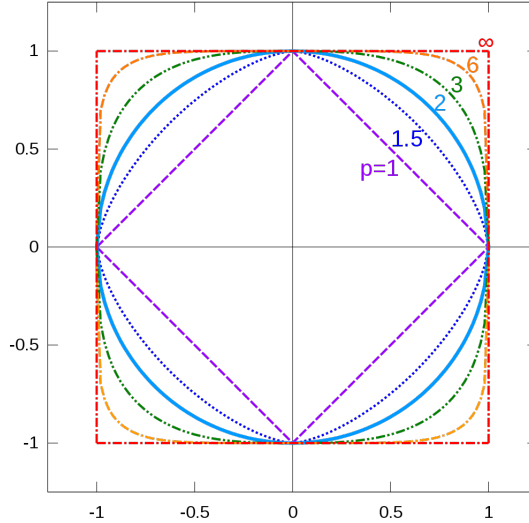


Figure 1.1. Unit balls for the ℓ_p norms in $\mathbb{R}^2$. [65]

The Fischer-Riesz theorem, stated next, guarantees that L^p spaces are always Banach spaces, securing the completeness needed for the perturbation analysis; the specialisation to $p = 2$ further yields a Hilbert space, whose additional inner-product structure is what makes cosine similarity well-defined. Both properties are examined in concrete terms in the next subsection.

Theorem 1.1.3 (Fischer-Riesz). $(L^p(X, \mu), \|\cdot\|_p)$ is a Banach space for every $p \in [1, +\infty]$.

Proof. See [12], Theorem 1.5-4 (completeness) and Theorem 2.2-3 (norm). $\square$

Corollary 1.1.1 (L^2 is a Hilbert space). $L^2(X, \mu)$ is a Hilbert space with inner product $\langle f, g \rangle := \int_X f g d\mu$. For $p \neq 2$, $L^p(X, \mu)$ does not admit an inner product compatible with $\|\cdot\|_p$.

In particular, $(\mathbb{R}^n, \|\cdot\|_p)$ is a Banach space for every p , and a Hilbert space if and only if $p = 2$, with inner product $\langle u, v \rangle := \sum_{i=1}^n u_i v_i$.

Finite-dimensional Normed Spaces

The previous subsection established the algebraic foundations of normed spaces valid for any vector space. I now specialise to the setting of this thesis: $\mathbb{R}^d$ with $d \in [768, 4096]$,

the codomain of every embedding model evaluated. Throughout this chapter, n denotes a generic dimension in abstract statements, while d refers specifically to this embedding codomain; definitions and theorems that hold for any finite-dimensional space are stated in $\mathbb{R}^n$, while those specific to the embedding setting use $\mathbb{R}^d$.

Finite dimension brings two consequences that are absent in the general case and that shape the entire experimental design. The first, developed in Theorem 1.1.4 and Lemma 1.1.4, is the *equivalence of all norms*: Lipschitz continuity of embedding function E in one norm implies it in all others, yet the equivalence constants grow with d , making the choice of output distance a genuine empirical question. The second, examined in the closing paragraphs of this subsection, arises from the *high dimensionality* of $\mathbb{R}^d$: concentration phenomena degrade distance-based reasoning and explain why cosine similarity emerges as the dominant retrieval metric. The proofs in this subsection follow [12, 13].

To prove the equivalence of norms, I first need a compactness result specific to finite-dimensional spaces. The key property, which fails in infinite dimension, is that closed and bounded sets are compact: this allows one to minimise continuous functions over spheres and extract the constants that appear in Lipschitz and equivalence bounds.

Definition 1.1.14 (Heine-Borel property [13, 1.8(i)]). *A topological vector space X has the Heine-Borel property if every closed and bounded subset of X is compact.*

Remark 1.1.4. $(\mathbb{R}^n, \|\cdot\|_2)$ has the Heine-Borel property: every subset of $\mathbb{R}^n$ that is closed and bounded with respect to the Euclidean norm is compact.

The Heine-Borel property allows one to extract bounded constants from optimisation over spheres. As a first application, I show that in finite dimension linearity alone forces Lipschitz continuity, motivating the use of the Lipschitz constant as a natural stability measure for embedding function E .

Corollary 1.1.2 (Linear maps are Lipschitz). *Every linear map $T : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is Lipschitz continuous, with $\text{Lip}(T) \leq \|T\|_{\text{op}}$, where $\|T\|_{\text{op}} := \sup_{\|v\|_2=1} \|Tv\|_2$.*

Proof. For any $u, v \in \mathbb{R}^n$, $\|Tu - Tv\|_2 = \|T(u - v)\|_2 \leq \|T\|_{\text{op}}\|u - v\|_2$. Finiteness of $\|T\|_{\text{op}}$ follows from compactness of $\mathbb{S}^{n-1}$ (closed and bounded in $\mathbb{R}^n$) and continuity of $v \mapsto \|Tv\|_2$. $\square$

Remark 1.1.5. *Embedding function E is in general non-linear, so Corollary 1.1.2 does not apply directly. It does, however, confirm that the Lipschitz constant is the right stability measure in finite dimension: any linear approximation of embedding function E would automatically satisfy a Lipschitz bound, and the question is whether E itself does so globally.*

I am now ready to state and prove the central structural result of this subsection. The same compactness argument used above, minimising a continuous function over the compact unit sphere, yields the equivalence constants for any two norms.

Theorem 1.1.4 (Equivalence of norms). *All norms on a finite-dimensional real vector space are equivalent: for any two norms $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$ on $\mathbb{R}^n$ there exist constants $c, C > 0$ such that*

$$c \|v\|_\alpha \leq \|v\|_\beta \leq C \|v\|_\alpha \quad \forall v \in \mathbb{R}^n.$$

Proof. It suffices to show that every norm $\|\cdot\|$ on $\mathbb{R}^n$ is equivalent to the Euclidean norm $\|\cdot\|_2$.

- **Step 1 (upper bound).** Let $e_1, \dots, e_n$ be the standard basis. For any $v \in \mathbb{R}^n$, the triangle inequality and Cauchy-Schwarz give

$$\|v\| = \left\| \sum_{j=1}^n v_j e_j \right\| \leq \sum_{j=1}^n |v_j| \|e_j\| \leq \|v\|_2 \left(\sum_{j=1}^n \|e_j\|^2 \right)^{1/2} =: C \|v\|_2.$$

- **Step 2 (lower bound).** The function $f(v) := \|v\|$ is continuous on $(\mathbb{R}^n, \|\cdot\|_2)$ by Step 1. The unit sphere $\mathbb{S}^{n-1}$ is compact (Remark 1.1.4). By the Weierstrass theorem, f attains its minimum $c := \min_{\|v\|_2=1} \|v\| > 0$ on $\mathbb{S}^{n-1}$. For any $v \neq 0$, applying this to $\hat{v} := v/\|v\|_2$ gives $\|v\| = \|v\|_2 \cdot \|\hat{v}\| \geq c \|v\|_2$.

Combining Steps 1 and 2 and using transitivity gives the claim for any two norms. $\square$

Norm equivalence has an important but easily misread implication for this thesis. On one hand, it guarantees that the *existence* of a finite Lipschitz constant for embedding function E is norm-independent: if E is Lipschitz in one norm, it is Lipschitz in all others, and the experimental design of Chapter 3 will therefore yield qualitatively concordant conclusions. On the other hand, topological equivalence does not imply numerical equivalence: the constants c and C depend on both the pair of norms and on the dimension d . The following lemma makes this dependence explicit for the most relevant pair.

Lemma 1.1.4 (ℓ_1 – ℓ_2 equivalence). *For all $v \in \mathbb{R}^n$,*

$$\|v\|_2 \leq \|v\|_1 \leq \sqrt{n} \|v\|_2.$$

Proof. The first inequality: $\|v\|_2^2 = \sum v_i^2 \leq (\sum |v_i|)^2 = \|v\|_1^2$. For the second, apply Cauchy-Schwarz to $(|v_1|, \dots, |v_n|)$ and $(1, \dots, 1)$: $\|v\|_1 \leq \sqrt{\sum v_i^2} \cdot \sqrt{n} = \sqrt{n} \|v\|_2$. $\square$

With $d \in [768, 4096]$, Lemma 1.1.4 gives a multiplicative gap of up to $\sqrt{d} \approx 55$ between the ℓ_1 and ℓ_2 norms. The analogous bounds for the other pairs are $\|v\|_\infty \leq \|v\|_2 \leq \sqrt{d} \|v\|_\infty$ and $\|v\|_\infty \leq \|v\|_1 \leq d \|v\|_\infty$. The upshot is that the *values* of all stability indices defined in Chapter 3 depend substantially on the choice of output norm, even when their qualitative behaviour agrees. Identifying which metric best captures the embedding displacement under typographical perturbations is therefore a genuine empirical question, and one of the main goals of Chapter 4.

The following theorem completes the structural picture by establishing that finite-dimensional normed spaces are always complete. Together with Theorem 1.1.4, it

guarantees that $(\mathbb{R}^d, \|\cdot\|)$ is a Banach space for every norm, providing the stable analytic framework required for the perturbation analysis.

Theorem 1.1.5 (Completeness of finite-dimensional spaces). *Every finite-dimensional normed space is complete, and hence a Banach space.*

Proof. It suffices to prove the claim for $(\mathbb{R}^n, \|\cdot\|_\infty)$. A sequence $(v^{(k)})$ is Cauchy for $\|\cdot\|_\infty$ if and only if each coordinate sequence $(v_i^{(k)})_{k \geq 1}$ is Cauchy in $(\mathbb{R}, |\cdot|)$. Since $\mathbb{R}$ is complete, each coordinate converges, and the limit vector is the limit in $(\mathbb{R}^n, \|\cdot\|_\infty)$. Completeness for any other norm then follows from Theorem 1.1.4. $\square$

With equivalence and completeness established, I can now identify the additional structure that distinguishes ℓ_2 from the other three norms. The Banach property, guaranteed for all four distances by Theorem 1.1.5, ensures that limits of Cauchy sequences of embedding vectors are well-defined, providing the stable framework required for the perturbation analysis of Chapter 3. The Hilbert property of $(\mathbb{R}^d, \|\cdot\|_2)$, by contrast, is exclusive to $p = 2$: for $p \neq 2$ no inner product is compatible with $\|\cdot\|_p$ (Corollary 1.1.1), so angle-based quantities are not definable from those norms. I therefore introduce cosine similarity, which is well-posed precisely because of this additional structure.

Definition 1.1.15 (Cosine similarity). *Given two non-zero vectors $u, v \in \mathbb{R}^d$, the cosine similarity is*

$$\cos(u, v) := \frac{\langle u, v \rangle}{\|u\|_2 \|v\|_2} \in [-1, 1].$$

Cosine similarity measures the angle between vectors, not their Euclidean separation, and is invariant under positive rescaling. This scale-invariance is the key reason cosine is the standard retrieval metric for embedding models: contrastive training with the InfoNCE loss (Section 2.1) encourages the model to encode semantic content in the *direction* of the output vector rather than in its magnitude. The derived distance is defined next.

Definition 1.1.16 (Cosine distance). *Given two non-zero vectors $u, v \in \mathbb{R}^d$, the cosine distance is*

$$d_{\cos}(u, v) := 1 - \cos(u, v) = 1 - \frac{\langle u, v \rangle}{\|u\|_2 \|v\|_2} \in [0, 2].$$

Restricted to the unit sphere $\mathbb{S}^{d-1}$, where $\|u\|_2 = \|v\|_2 = 1$, $d_{\cos}$ is a metric and satisfies

$$\|u - v\|_2^2 = 2 d_{\cos}(u, v),$$

so nearest-neighbour retrieval under cosine distance is equivalent to nearest-neighbour retrieval under Euclidean distance after ℓ_2 -normalisation.

Remark 1.1.6 (Cosine distance on $\mathbb{S}^{d-1}$). *On all of $\mathbb{R}^d$, $d_{\cos}$ is not a metric: it is undefined at the origin, and fails non-degeneracy since $d_{\cos}(u, v) = 0$ whenever $u = \lambda v$ with $\lambda > 0$, even if $u \neq v$. These pathologies vanish on $\mathbb{S}^{d-1}$, where positive proportionality forces $\lambda = 1$ and hence $u = v$, recovering non-degeneracy.*

Remark 1.1.7. *All distance metrics in this thesis are computed directly on the embedding vectors as returned by each provider, without additional ℓ_2 -normalisation. Several providers approximately normalise their output; when this holds exactly, cosine and Euclidean distances are monotonically related by $\|u - v\|_2^2 = 2 d_{\cos}(u, v)$ and induce the same ordering. In practice, normalisation is imperfect or absent for some of the six embedding models evaluated, so the two metrics are treated as distinct throughout the experiments of Chapter 4. This equivalence is in any case specific to the cosine–Euclidean pair: the ℓ_1 and ℓ_∞ distances between unit vectors depend on the full coordinate structure and cannot be expressed in terms of the inner product alone, so they measure genuinely different geometric quantities even on $\mathbb{S}^{d-1}$. The differential sensitivity of all four metrics to typographical perturbations is one of the empirical questions addressed in Chapter 4.*

Curse of dimensionality. In high-dimensional spaces a well-known phenomenon degrades the utility of distance-based reasoning: as dimension $d \rightarrow \infty$, the ratio between the maximum and minimum pairwise distances among N random points concentrates around 1 [22]. Formally, letting $x_1, \dots, x_N \in \mathbb{R}^d$ be i.i.d. random points and $i, j \in \{1, \dots, N\}$ with $i \neq j$ index pairs among them,

$$\lim_{d \rightarrow \infty} \frac{\max_{i,j} \|x_i - x_j\|_p - \min_{i,j} \|x_i - x_j\|_p}{\min_{i,j} \|x_i - x_j\|_p} = 0.$$

This *distance concentration* implies that nearest-neighbour queries lose discriminability: all points appear approximately equidistant, making the notion of “closest vector” unreliable. The phenomenon is most severe for $p < 2$ and attenuates as $p \rightarrow \infty$ [22]. Among the four distances considered in this thesis, Chebyshev ($p = \infty$) is therefore the most robust against distance concentration in theory; cosine, by contrast, is insensitive to magnitude and measures only angular separation, which remains discriminative even as d grows, because it operates on directions rather than absolute coordinate values.

A natural question is whether the curse is alleviated by the geometry of the embedding output space. Embedding vectors do not fill $\mathbb{R}^d$ uniformly: the image of embedding function E concentrates on a lower-dimensional structure whose intrinsic dimension is governed by the data, not by the ambient space. If the effective geometry is that of a submanifold of intrinsic dimension $k \ll d$, then distance concentration is governed by k rather than d , and the curse is attenuated rather than eliminated. The intrinsic dimension of the latent manifold of BERT-style encoders is estimated to be in the low hundreds [23], substantially lower than $d \in [768, 4096]$: pairwise distances are therefore more discriminative than they would be for points distributed uniformly in $\mathbb{R}^d$, yet still more concentrated than in a low-dimensional flat space.

This observation motivates the geometric framework developed in the next subsection. The fact that embedding vectors concentrate on a curved lower-dimensional structure rather than filling $\mathbb{R}^d$ calls for a mathematical language capable of describing the intrinsic geometry of such structures: the theory of Riemannian manifolds.

Riemannian Manifolds

Finite-dimensional normed spaces such as $(\mathbb{R}^d, \|\cdot\|_2)$ provide a flat, globally uniform geometry. In the context of embedding models, however, the output vectors do not fill $\mathbb{R}^d$ uniformly: the image of embedding function E concentrates on a much lower-dimensional subspace whose geometry reflects the structure of the data. With ℓ_2 -normalisation, this subspace is exactly the unit sphere $\mathbb{S}^{d-1} \subset \mathbb{R}^d$. Understanding the intrinsic geometry of this curved subspace requires a generalisation of the flat inner-product geometry of $\mathbb{R}^d$: the theory of Riemannian manifolds, introduced in this subsection.

The starting point is the notion of a smooth manifold, which captures the idea of a space that is locally flat but globally curved.

Definition 1.1.17 (Smooth manifold). *A smooth manifold of dimension k is a topological space $\mathcal{M}$ such that every point $p \in \mathcal{M}$ has an open neighbourhood homeomorphic to $\mathbb{R}^k$, with smooth transition maps between overlapping neighbourhoods. The integer k is the dimension of $\mathcal{M}$.*

Intuitively, a smooth manifold looks like $\mathbb{R}^k$ when viewed up close at any single point, but may have a globally curved shape. A surface embedded in $\mathbb{R}^3$, such as a sphere or a torus, is the prototypical example. To measure distances and angles on such a surface, one needs to assign a notion of inner product at each point; this is precisely what a Riemannian metric provides.

Definition 1.1.18 (Riemannian manifold [14]). *A Riemannian manifold is a pair $(\mathcal{M}, g)$ where $\mathcal{M}$ is a smooth manifold and g is a Riemannian metric: a smooth assignment $p \mapsto g_p$, where*

$$g_p : T_p\mathcal{M} \times T_p\mathcal{M} \rightarrow \mathbb{R}$$

is a symmetric, positive definite bilinear form on the tangent space $T_p\mathcal{M}$ at each point p . The metric g varies smoothly with p and defines lengths of tangent vectors, angles between them, and lengths of curves on $\mathcal{M}$.

Remark 1.1.8. *The tangent space $T_p\mathcal{M}$ at a point p is the linear space of all directional derivatives at p : it is the best linear approximation of $\mathcal{M}$ near p . For a surface embedded in $\mathbb{R}^3$, $T_p\mathcal{M}$ is simply the plane tangent to the surface at p . The Riemannian metric g_p then plays the role of an inner product on this tangent plane, allowing one to measure the length of curves on $\mathcal{M}$ by integrating infinitesimal tangent vectors.*

The most relevant instance for this thesis is the unit sphere, which arises directly from ℓ_2 -normalisation of embedding vectors.

Example 1.1.3 (The unit sphere). *The unit sphere $\mathbb{S}^{d-1} := \{v \in \mathbb{R}^d : \|v\|_2 = 1\}$ is a Riemannian manifold of dimension $d - 1$, embedded in $\mathbb{R}^d$. At each point $p \in \mathbb{S}^{d-1}$, the tangent space is*

$$T_p\mathbb{S}^{d-1} = \{v \in \mathbb{R}^d : \langle v, p \rangle = 0\},$$

the hyperplane orthogonal to p . The Riemannian metric is the restriction of the Euclidean inner product of $\mathbb{R}^d$ to each such hyperplane. Globally, $\mathbb{S}^{d-1}$ is curved: the shortest path

(geodesic) between two points is an arc of great circle, not the Euclidean straight line through the interior of the ball, as illustrated in Figure 1.2.

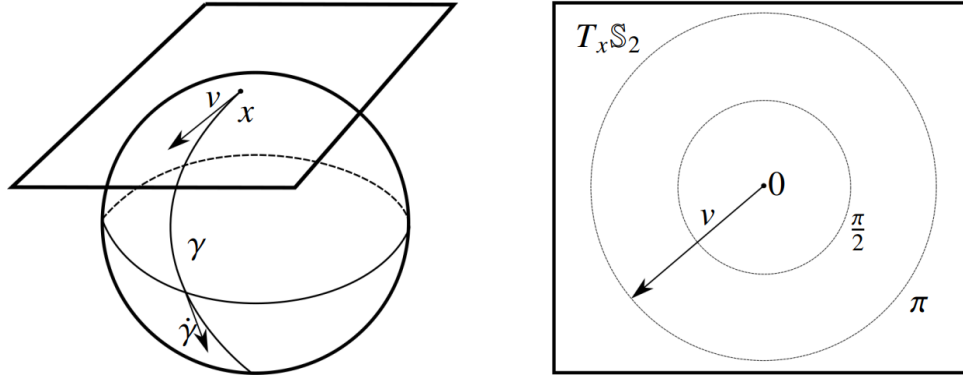


Figure 1.2. The unit sphere $\mathbb{S}^2 \subset \mathbb{R}^3$ with tangent plane $T_x \mathbb{S}^2$ at a point x . The geodesic γ follows an arc of great circle on the surface, not the Euclidean straight line through the interior [66].

With the output space of embedding function E now characterised both algebraically and geometrically, I turn to the input side: the space Σ^* of strings, equipped with the edit distances that measure typographical proximity between queries.

1.2 Edit Distances

Edit distances provide a fundamental framework for measuring the dissimilarity between strings by quantifying the minimum cost required to transform one string into another. They are a central object in approximate string matching, with extensive applications in text processing, computational biology [24], information retrieval [15, 17], and clinical pathway mining [67].

String Space and Approximate Matching

Let Σ be a finite alphabet of size $|\Sigma| = \sigma$. I denote by Σ^* the set of all finite strings over Σ , including the empty string ε . For any string $s \in \Sigma^*$, its length is denoted $|s|$, its i -th character is written s_i for $i \in \{1, \dots, |s|\}$, and the substring $s_{i..j} = s_i s_{i+1} \dots s_j$ denotes the segment from position i to position j (the empty string if $i > j$).

Following Navarro [15], a *pattern* P is a short reference string that I wish to locate, possibly with errors, inside a longer *text* T . The classical *approximate string matching* problem is formulated as follows. Let

- $T \in \Sigma^*$ be a *text* of length $n := |T|$;
- $P \in \Sigma^*$ be a *pattern* of length $m := |P|$;

- $k \in \mathbb{R}_{\geq 0}$ be the maximum allowed number of errors;
- $d : \Sigma^* \times \Sigma^* \rightarrow \mathbb{R}_{\geq 0} \cup \{\infty\}$ be a distance function.

The goal is to return all positions j such that $d(P, T_{i..j}) \leq k$ for some $i \leq j$. The quantity $\alpha = k/m \in (0, 1)$ is called the *error level*; reasonable values in practice range from $1/m$ to $1/2$.

In this thesis, the text-pattern pair is replaced by the following setting:

- $s \in \Sigma^*$ is a *clean query* of length $m = |s|$;
- $\tilde{s} \in \Sigma^*$ is a *perturbed variant* of s , obtained by applying a controlled sequence of edit operations to s ;
- $k := d(s, \tilde{s}) \in \mathbb{R}_{\geq 0}$ is the edit distance between the two strings, measuring the intensity of the perturbation.

The error level $\alpha = k/m$ provides a normalised measure of perturbation intensity, comparable across queries of different lengths.

The central question of this thesis is then: does the embedding model E respond to typographical perturbations in a metrically stable way? Formally, I study the displacement $\|E(s) - E(\tilde{s})\|$ as a function of the edit distance $d(s, \tilde{s})$, asking whether E acts as a Lipschitz map from (Σ^*, d) into $(\mathbb{R}^d, \|\cdot\|)$ and, when it does not, characterising the scale and structure of the distortion.

General edit distance properties

The general formulation of edit distance is based on a set of allowed operations, each associated with a non-negative cost.

Definition 1.2.1 (General edit distance). *Let Σ^* denote the set of all finite strings over an alphabet Σ . Fix a finite family $\mathcal{O}$ of elementary operations, where each operation $o \in \mathcal{O}$ replaces a substring $z \in \Sigma^*$ by a string $w \in \Sigma^*$ at a non-negative cost $c(o) \in \mathbb{R}_{\geq 0}$.*

The edit distance $d(s, t)$ between two strings $s, t \in \Sigma^$ is the minimum total cost over all finite sequences of elementary operations transforming s into t :*

$$d(s, t) := \inf \left\{ \sum_k c(o_k) \mid o_1, \dots, o_m \in \mathcal{O}, o_m \circ \dots \circ o_1(s) = t \right\}.$$

If no such sequence exists, $d(s, t) = +\infty$.

A finite sequence of elementary operations $(o_1, \dots, o_m)$ such that $o_m \circ \dots \circ o_1(s) = t$ is called an *edit sequence* from s to t ; its total cost is $\sum_{k=1}^m c(o_k)$. In the computational biology literature, an analogous object is often called an *alignment*; I reserve this term for sequence alignment problems [24] and use *edit sequence* throughout.

In most applications, $\mathcal{O}$ is restricted to the following *single-character elementary operations*, each identified by a fixed symbol used throughout this section:

- *Insertion* $\iota(b)$ of character $b \in \Sigma$: replace the empty string ε by b , at cost $c_\iota(b) \geq 0$;
- *Deletion* $\delta(a)$ of character $a \in \Sigma$: replace a by ε , at cost $c_\delta(a) \geq 0$;
- *Substitution* $\sigma(a, b)$ of a by b ($a \neq b$): replace a by b , at cost $c_\sigma(a, b) \geq 0$;
- *Transposition* $\tau(a, b)$ of adjacent characters ab ($a \neq b$): replace ab by ba , at cost $c_\tau(a, b) \geq 0$.

For example, transforming **kitten** into **sitting** requires $\sigma(k, s)$, $\sigma(e, i)$, and $\iota(g)$ at the end, for a total cost of 3 under unit costs.

Some formulations of edit distance, including the one adopted by Navarro [15], add the following restriction to Definition 1.2.1: *once an elementary operation has converted a substring z into w , no further operation may act on the characters of w* . This is called the *no-reuse* (or *restricted*) condition.

The restriction is motivated by computability. Without it, alternating inverse operations (such as $\sigma(a, b)$ followed by $\sigma(b, a)$) produce edit sequences of unbounded length, making it undecidable whether $d(s, t) \leq k$ in general. Requiring each output substring to be modified at most once bounds the length of any admissible sequence by $|s| + |t|$, ensuring computability.

When $\mathcal{O}$ contains only insertions, deletions, and substitutions, the no-reuse condition is automatically satisfied: since each of these operations acts on a single character and produces a single character (or the empty string), no operation can produce a substring that a later operation could usefully target again. The condition therefore does not affect the distance values in this case. The situation changes when transpositions are admitted, as discussed in Section 1.2; it is in that context that the restricted and unrestricted models diverge. Note, however, that edit distances with transpositions can still be computed efficiently via dynamic programming regardless of whether the restriction is imposed.

Under mild conditions on the cost model, edit distances are genuine metrics on Σ^* , as the following theorem establishes.

Theorem 1.2.1 (Edit distances are metrics). *Let d be an edit distance as in Definition 1.2.1. If all elementary operations have strictly positive cost and every operation has an inverse of equal cost, then d is a metric on Σ^* .*

Proof. Since Σ is finite, the set of elementary operation costs is finite, and any sequence cost is a finite sum of these costs with non-negative integer coefficients. The infimum over all sequences from s to t is therefore attained: a minimum-cost sequence always exists [25]. The three metric axioms of Definition 1.1.1 are satisfied:

- Non-degeneracy.** Every sequence of operations has non-negative total cost, so $d(s, t) \geq 0$. The empty sequence transforms s into itself at cost 0, so $d(s, s) = 0$. Conversely, if $d(s, t) = 0$ for $s \neq t$, any optimal sequence has total cost 0; since all costs are strictly positive, the sequence must be empty, but an empty sequence leaves the string unchanged, so $s = t$, a contradiction.

- (b) **Symmetry.** By hypothesis, every elementary operation has an inverse of equal cost. Given an optimal edit sequence $(o_1, \dots, o_m)$ from s to t , the reversed sequence of inverses $(o_m^{-1}, \dots, o_1^{-1})$ is a valid edit sequence from t to s of equal total cost. Hence $d(s, t) = d(t, s)$.
- (c) **Triangle inequality.** Let $(o_1, \dots, o_m)$ be optimal from s to t , and $(p_1, \dots, p_k)$ optimal from t to u . Their concatenation is a valid edit sequence from s to u of cost $d(s, t) + d(t, u)$. Since $d(s, u)$ is the minimum,

$$d(s, u) \leq d(s, t) + d(t, u).$$

□

The metric property established above immediately implies that the string space (Σ^*, d) is complete whenever the operation costs are integer-valued, by the same argument used for discrete spaces in Proposition 1.1.1.

Proposition 1.2.1 (Completeness of string spaces). *Let d be an edit distance as in Definition 1.2.1 with positive integer operation costs. Then (Σ^*, d) is complete.*

Proof. Under the stated assumption, d takes values in $\mathbb{N}_{\geq 0}$. The claim then follows immediately from Proposition 1.1.1. □

Remark 1.2.1. *The assumption that costs are integers is satisfied in particular by unit-cost edit distances such as the ones presented in Section 1.2. If costs are allowed to be arbitrary positive reals, the argument breaks down.*

Completeness ensures that (Σ^*, d) is a well-behaved metric space, but a further structural observation is needed before turning to the specific distances used in this thesis. The following proposition shows that every function out of (Σ^*, d) is automatically continuous, which makes ordinary continuity vacuous as a stability criterion and motivates the use of Lipschitz conditions instead.

Proposition 1.2.2 (Discrete topology induced by edit distances). *Let Σ be a finite alphabet, d a unit-cost edit distance on Σ^* , and $s \in \Sigma^*$ any string. Then for every real number $r \in (0, 1)$, the open ball of radius r centred at s is the singleton*

$$B(s, r) = \{s\}.$$

In particular, every singleton $\{s\} \subseteq \Sigma^$ is open, the topology induced by d on Σ^* is discrete, and every function $f: (\Sigma^*, d) \rightarrow (Y, d_Y)$ into an arbitrary metric space (Y, d_Y) is continuous.*

Proof. For any $t \neq s$, at least one edit operation is required to transform s into t , so $d(s, t) \geq 1$. For any $r \in (0, 1)$ one therefore has

$$B(s, r) = \{t \in \Sigma^* : d(s, t) < r\} = \{s\}.$$

Since every singleton is open, the induced topology is discrete. For any map f and any $\varepsilon > 0$, choose $\eta = \frac{1}{2}$: if $d(s, t) < \eta$ then $t = s$, hence $d_Y(f(s), f(t)) = 0 < \varepsilon$. Thus f is continuous at every point. □

Remark 1.2.2. *This proposition shows that continuity is too weak a notion for string spaces: although every map $(\Sigma^*, d) \rightarrow (Y, d_Y)$ is automatically continuous, it places no bound on the output variation when the input changes by a single edit. Lipschitz conditions are therefore the appropriate stability requirement when studying embedding models.*

Main Edit Distances

I now recall the four most common edit distances obtained from different subsets of the operations in Definition 1.2.1, following the classification of [15]. For each distance, I verify the hypotheses of Theorem 1.2.1: strictly positive operation costs, and the existence of an inverse operation of equal cost within the class of admissible sequences.

Definition 1.2.2 (Hamming distance). *The Hamming distance $d_H(s, t)$ between two strings $s, t \in \Sigma^*$ of equal length $|s| = |t| = n$ is the number of positions at which they differ:*

$$d_H(s, t) := \#\{i \in [n] : s_i \neq t_i\},$$

with $0 \leq d_H(s, t) \leq |s|$.

Example 1.2.1. $d_H(\text{karolin}, \text{kathrin}) = 3$: positions 3, 4, and 5 differ.

The Hamming distance is the most restrictive of the four: it operates only on strings of equal length and admits no insertions or deletions. Relaxing the length constraint while retaining only insertions and deletions yields the LCS distance.

Definition 1.2.3 (LCS distance). *The Longest Common Subsequence distance (LCS) allows only unit-cost insertions and deletions. If $\text{LCS}(s, t)$ denotes the length of the longest common subsequence of $s, t \in \Sigma^*$, then*

$$d_{\text{LCS}}(s, t) := |s| + |t| - 2 \text{LCS}(s, t),$$

with $0 \leq d_{\text{LCS}}(s, t) \leq |s| + |t|$.

Example 1.2.2. For $s = \text{ABCB DAB}$ and $t = \text{BDCABA}$, the LCS has length 4 (BCBA), giving $d_{\text{LCS}}(s, t) = 7 + 6 - 8 = 5$.

Adding substitutions to insertions and deletions gives the most widely used edit distance in natural language processing, the Levenshtein distance.

Definition 1.2.4 (Levenshtein distance). *The Levenshtein distance $d_L(s, t)$ is the minimum number of unit-cost insertions ι , deletions δ , and substitutions σ required to transform s into t , with $0 \leq d_L(s, t) \leq \max(|s|, |t|)$.*

Example 1.2.3. $d_L(\text{kitten}, \text{sitting}) = 3$, achieved by $\sigma(k, s)$, $\sigma(e, i)$, and $\iota(g)$ at the end.

All three distances defined so far satisfy the hypotheses of Theorem 1.2.1 and are therefore genuine metrics.

Proposition 1.2.3 (Hamming, LCS, Levenshtein are metrics). *The Hamming, LCS, and Levenshtein distances are metrics on their respective domains.*

Proof. All three distances involve only the operations $\iota(b)$, $\delta(a)$, and $\sigma(a, b)$. Since each acts on a single character, both hypotheses of Theorem 1.2.1 are satisfied: all operations have unit cost, and each admits an inverse of equal cost ($\iota(b)$ is inverted by $\delta(b)$ and vice versa; $\sigma(a, b)$ is inverted by $\sigma(b, a)$). $\square$

I now introduce the two distances most relevant to this thesis, which extend Levenshtein by admitting the transposition $\tau(a, b)$, but differ in how they handle the no-reuse restriction.

For ι , δ , and σ , the restriction is inert: each operation acts on a single character and produces a substring of length at most 1, so no subsequent operation can act inside it. The transposition $\tau(a, b)$, however, produces a two-character output ba , and a subsequent operation may need to act on one of those characters, which the restricted model forbids. To see why this matters concretely, consider transforming ca into abc . The natural two-step path applies $\tau(c, a)$ to obtain ac , then inserts b between a and c via $\iota(b)$ to obtain abc . Under the no-reuse restriction, however, this path is inadmissible: because ac was produced by $\tau(c, a)$, no further operation may act on its characters, and in particular $\iota(b)$ cannot insert between them. The restricted model therefore forces a longer route of cost 3, while the unrestricted model accepts the two-step path at cost 2. This divergence affects the metric properties of the resulting distance, as shown in the remainder of this section.

Definition 1.2.5 (Optimal String Alignment distance). *The Optimal String Alignment (OSA) distance is the restricted variant: it admits transpositions but enforces the no-reuse condition, so that once a transposition $\tau(a, b)$ has produced the pair ba , no further operation may act on those two characters within the same edit sequence. Formally, the OSA distance between $s, t \in \Sigma^*$ is defined via the recurrence*

$$d_{s,t}(i, j) := \min \begin{cases} 0 & \text{if } i = j = 0, \\ d_{s,t}(i-1, j) + 1 & \text{if } i > 0, \\ d_{s,t}(i, j-1) + 1 & \text{if } j > 0, \\ d_{s,t}(i-1, j-1) + \mathbf{1}_{[s_i \neq t_j]} & \text{if } i, j > 0, \\ d_{s,t}(i-2, j-2) + \mathbf{1}_{[s_i \neq t_j]} & \text{if } i, j > 1, s_i = t_{j-1}, s_{i-1} = t_j, \end{cases}$$

where $d_{s,t}(i, j)$ is the OSA distance between the prefixes $s_{1..i}$ and $t_{1..j}$, and $\mathbf{1}_{[s_i \neq t_j]}$ equals 1 if $s_i \neq t_j$ and 0 otherwise. The five branches correspond, in order, to the base case, $\delta(s_i)$, $\iota(t_j)$, match or $\sigma(s_i, t_j)$, and $\tau(s_{i-1}, s_i)$. The no-reuse constraint is encoded in the transposition branch by the fixed look-back of exactly two positions: a character involved in a transposition at step $(i-2, j-2)$ cannot participate in any subsequent branch of the same recurrence path. The OSA distance is $d_{\text{OSA}}(s, t) = d_{s,t}(|s|, |t|)$.

Example 1.2.4. Consider $s = \mathbf{abdc}$ and $t = \mathbf{abcd}$, which differ only in that the last two characters are swapped. The Levenshtein distance is $d_L(\mathbf{abdc}, \mathbf{abcd}) = 2$ (substitute

$d \rightarrow c$ at position 3 and $c \rightarrow d$ at position 4, or equivalently delete one and reinsert). The OSA distance, by contrast, is $d_{\text{OSA}}(\mathbf{abdc}, \mathbf{abcd}) = 1$, achieved by the single transposition $\tau(d, c)$ acting on the adjacent pair dc at positions 3-4. More generally, any two strings that differ by exactly one adjacent transposition of distinct characters have OSA distance 1 and Levenshtein distance 2.

Although the OSA distance correctly counts transpositions, the no-reuse restriction prevents it from satisfying the triangle inequality. The following remark identifies the precise point at which the argument of Theorem 1.2.1 breaks down, and the subsequent proposition confirms the failure with an explicit counterexample.

Remark 1.2.3 (Why OSA fails Theorem 1.2.1). *The hypothesis of Theorem 1.2.1 concerning inverses requires not only that each operation have a formal inverse of equal cost, but that this inverse be executable within the same class of admissible sequences. For OSA, this fails. The transposition $\tau(a, b)$ is formally self-inverse, but its application “consumes” the pair: the no-reuse restriction in Definition 1.2.5 forbids any further operation on the same substring within the same sequence. As a consequence, the class of admissible OSA sequences is not closed under concatenation: if $\pi_1 : s \rightarrow t$ and $\pi_2 : t \rightarrow u$ are valid OSA sequences, their concatenation $\pi_2 \circ \pi_1$ may not be admissible, because a character transposed in π_1 may require re-editing in π_2 . Since the proof of the triangle inequality in Theorem 1.2.1 relies on exactly this concatenation to bound $d(s, u) \leq d(s, t) + d(t, u)$, the argument breaks down.*

Proposition 1.2.4 (OSA is not a metric). *The OSA distance does not satisfy the triangle inequality and is therefore not a metric on Σ^* .*

Proof. Let $s_1 = \mathbf{ca}$, $s_2 = \mathbf{ac}$, $s_3 = \mathbf{abc}$, where strings are written in monospace and individual characters are referred to by the corresponding abstract variables $a, b, c \in \Sigma$. One verifies that

$$d_{\text{OSA}}(s_1, s_2) = 1, \quad d_{\text{OSA}}(s_2, s_3) = 1, \quad d_{\text{OSA}}(s_1, s_3) = 3.$$

The first value follows from the single transposition $\tau(c, a)$, which swaps the two characters of $\mathbf{ca}$ to yield $\mathbf{ac}$. The second follows from the single insertion $\iota(b)$ between a and c in $\mathbf{ac}$ to yield $\mathbf{abc}$. For the third, the no-reuse restriction makes the two-step path $\mathbf{ca} \xrightarrow{\tau(c,a)} \mathbf{ac} \xrightarrow{\iota(b)} \mathbf{abc}$ inadmissible: once $\tau(c, a)$ has produced $\mathbf{ac}$, neither a nor c may be acted upon further, so $\iota(b)$ cannot insert between them. The shortest admissible path from s_1 to s_3 is therefore $\mathbf{ca} \xrightarrow{\delta(c)} \mathbf{a} \xrightarrow{\iota(b)} \mathbf{ab} \xrightarrow{\iota(c)} \mathbf{abc}$, at cost 3. Consequently,

$$d_{\text{OSA}}(s_1, s_2) + d_{\text{OSA}}(s_2, s_3) = 2 < 3 = d_{\text{OSA}}(s_1, s_3),$$

in violation of the triangle inequality. □

To recover the metric property while retaining transpositions, it suffices to lift the no-reuse restriction, which yields the Damerau-Levenshtein distance.

Definition 1.2.6 (Damerau-Levenshtein distance). *The Damerau-Levenshtein distance $d_{DL}(s, t)$ is defined via the auxiliary function*

$$d_{s,t}(i, j) := \min \begin{cases} 0 & \text{if } i = j = 0, \\ d_{s,t}(i-1, j) + 1 & \text{if } i > 0, \\ d_{s,t}(i, j-1) + 1 & \text{if } j > 0, \\ d_{s,t}(i-1, j-1) + \mathbf{1}_{[s_i \neq t_j]} & \text{if } i, j > 0, \\ d_{s,t}(i'-1, j'-1) + (i-i'-1) + (j-j'-1) + 1 & \text{if } i, j > 1, s_i = t_{j'}, s_{i'} = t_j, \end{cases}$$

where

$$\begin{aligned} i' &:= \max\{i'' < i \mid s_{i''} = t_j\}, \\ j' &:= \max\{j'' < j \mid t_{j''} = s_i\} \end{aligned}$$

are the last positions in s and t where the two crossing characters previously appeared. The distance between the full strings is $d_{DL}(s, t) = d_{s,t}(|s|, |t|)$.

Remark 1.2.4. *The first four branches of the Damerau-Levenshtein recurrence coincide with those of OSA, corresponding to the base case, $\delta(s_i)$, $\iota(t_j)$, and match or $\sigma(s_i, t_j)$. The last branch models an unrestricted transposition τ : it locates the last positions i' and j' where the crossing characters appeared, and accounts for the $(i-i'-1)$ deletions and $(j-j'-1)$ insertions needed to bring them adjacent, plus 1 for the transposition itself. The two recurrences differ only in this transposition branch: in OSA the look-back is fixed at exactly two positions, while in DL the indices i' and j' scan all previous matches, allowing τ to interact with earlier edits.*

As a direct consequence, $d_{DL}(s, t) \leq d_{OSA}(s, t)$ for every $s, t \in \Sigma^*$; in the proof of Proposition 1.2.4, this is witnessed concretely by $d_{DL}(s_1, s_3) = 2 < 3 = d_{OSA}(s_1, s_3)$.

Example 1.2.5. *With $s_1 = ca$ and $s_3 = abc$ as above, $d_{DL}(s_1, s_3) = 2$: the unrestricted model permits $\iota(b)$ after $\tau(c, a)$, a two-step path that OSA forbids.*

With the no-reuse restriction lifted and the class of admissible sequences now closed under concatenation, all hypotheses of Theorem 1.2.1 are satisfied, and the metric property follows directly.

Theorem 1.2.2 (Damerau-Levenshtein is a metric). *The Damerau-Levenshtein distance d_{DL} is a metric on Σ^* .*

Proof. All four operations $\iota(b)$, $\delta(a)$, $\sigma(a, b)$, $\tau(a, b)$ have unit cost and admit inverses of equal cost: $\iota(b)$ is inverted by $\delta(b)$ and vice versa; $\sigma(a, b)$ is inverted by $\sigma(b, a)$; and $\tau(a, b)$ is its own inverse. Since the no-reuse restriction is lifted, the class of admissible DL sequences is closed under concatenation. Theorem 1.2.1 then guarantees all metric axioms. $\square$

The Damerau-Levenshtein distance is particularly well-suited for modelling typographical errors: Damerau’s original study reported that over 80% of spelling mistakes in an information retrieval system were correctable by a single one of the four operations [16].

I now turn to the probabilistic formalisation of these operations: the next section models character-level perturbations as transitions of a finite Markov chain, providing the stochastic framework used to generate controlled typographical variants of query strings throughout the experiments.

1.3 Markov Chains and Typographical Noise

The generation of typographical variants of a given string can be formalised probabilistically by means of finite-state Markov chains. This section recalls the basic theory and connects it to the noise model used in this thesis. The treatment follows [20, 18, 19].

Finite Markov Chains

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space. A *stochastic process* is a family $(X_n)_{n \geq 0}$ of random variables $X_n : \Omega \rightarrow I$, where I is a finite set called the *state space*; each element $i \in I$ is a *state*. The *law* of the process at time n is the probability vector $\mu_n \in \mathbb{R}^I$ defined by $(\mu_n)_i := \mathbb{P}(X_n = i)$; the vector μ_0 is the *initial distribution*.

Throughout this thesis, for any random variable $Z : \Omega \rightarrow S$ and any $s \in S$, I use the shorthand $\{Z = s\}$ for the event $\{\omega \in \Omega : Z(\omega) = s\}$. In particular, $\{X_n = i\}$, $\{T_j = n\}$, and $\{T_j < \infty\}$ are always understood as events in $(\Omega, \mathcal{F}, \mathbb{P})$.

The first structural requirement on the process is the Markov property, which ensures that the corruption of each character in a string depends only on that character and not on what preceded it, making the confusion matrix a well-defined stochastic object.

Definition 1.3.1 (Markov property). *A stochastic process $(X_n)_{n \geq 0}$ with values in I satisfies the Markov property if, for all $n \geq 0$ and all states $i_0, \dots, i_{n+1} \in I$ with $\mathbb{P}(X_n = i_n, \dots, X_0 = i_0) > 0$,*

$$\mathbb{P}(X_{n+1} = i_{n+1} \mid X_n = i_n, \dots, X_0 = i_0) = \mathbb{P}(X_{n+1} = i_{n+1} \mid X_n = i_n).$$

A process satisfying the Markov property is called a Markov chain on I .

The Markov property expresses *memorylessness*: the future evolution of the chain depends on the present state alone, not on the history of the process. The quantities

$$p_{ij}^{(n)} := \mathbb{P}(X_{n+1} = j \mid X_n = i), \quad i, j \in I,$$

are called *transition probabilities* at time n ; they may in principle depend on n , but in the setting of this thesis they do not. An illustration of both the Markov property and its failure is given in Example 1.3.1.

In the noise model of this thesis, the corruption rule applied to each character is the same regardless of the position in the string or the step of the iteration, so the relevant chains are time-homogeneous, as formalised next.

Definition 1.3.2 (Time-homogeneity). *A Markov chain $(X_n)_{n \geq 0}$ is time-homogeneous if the transition probabilities $p_{ij}^{(n)}$ do not depend on n , for every $i, j \in I$ with $\mathbb{P}(X_n = i) > 0$. In this case I write $p_{ij} := p_{ij}^{(n)}$.*

Remark 1.3.1 (Standing convention). *All Markov chains considered in this thesis are time-homogeneous. I drop this qualifier henceforth, writing p_{ij} for the transition probabilities.*

The transition probabilities can be collected into a single matrix, whose row-stochastic structure is the algebraic counterpart of the constraint that the chain must occupy some state at every step.

Definition 1.3.3 (Transition matrix). *The transition matrix of a Markov chain is the matrix $P = (p_{ij})_{i,j \in I}$ defined by*

$$p_{ij} := \begin{cases} \mathbb{P}(X_{n+1} = j \mid X_n = i) & \text{if } \mathbb{P}(X_n = i) \neq 0, \\ 1 & \text{if } \mathbb{P}(X_n = i) = 0 \text{ and } i = j, \\ 0 & \text{if } \mathbb{P}(X_n = i) = 0 \text{ and } i \neq j. \end{cases}$$

Proposition 1.3.1. *The transition matrix P is row-stochastic: its entries are non-negative and every row sums to 1,*

$$p_{ij} \geq 0 \quad \forall i, j \in I, \quad \sum_{j \in I} p_{ij} = 1 \quad \forall i \in I.$$

Proof. Non-negativity follows directly from p_{ij} being a probability. For the row-sum condition, fix i with $\mathbb{P}(X_n = i) > 0$; then

$$\sum_{j \in I} p_{ij} = \sum_{j \in I} \mathbb{P}(X_{n+1} = j \mid X_n = i) = \mathbb{P}\left(\bigcup_{j \in I} \{X_{n+1} = j\} \mid X_n = i\right) = 1,$$

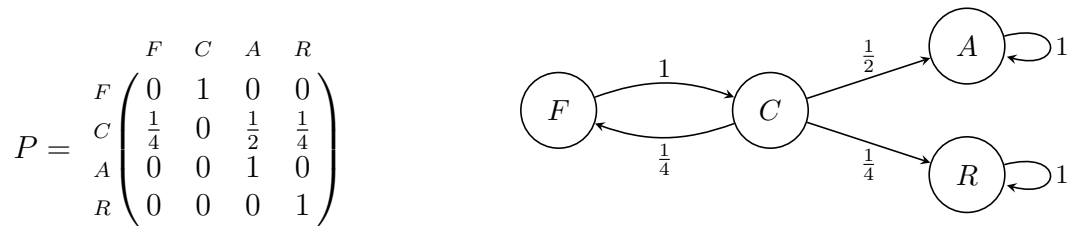
where the second equality uses the fact that the events $\{X_{n+1} = j\}_{j \in I}$ are pairwise disjoint, and the third uses the fact that they are exhaustive. The case $\mathbb{P}(X_n = i) = 0$ is covered by the convention $p_{ii} = 1$. $\square$

A Markov chain can equivalently be represented as a weighted directed graph $G = (V, E, w)$, where the node set $V = I$ contains the states, there is a directed edge $(i, j) \in E$ whenever $p_{ij} > 0$, and the weight of that edge is $w(i, j) = p_{ij}$. This *transition graph* makes the structure of the chain immediately visible: reachability in the graph corresponds to accessibility of states, and closed communicating classes correspond to strongly connected components with no outgoing edges. The transition matrix and graph of a running example are shown in Example 1.3.1.

Example 1.3.1 (The typo-correction pipeline). A spell-checker processes tokens through a pipeline with four states $I = \{F, C, A, R\}$: a token is either flagged for review (F), under correction (C), accepted (A), or rejected (R).

A natural but problematic routing rule is the following: a token under correction is accepted if it is being corrected for the first time, and rejected if it has already passed through correction before. This rule is non-Markovian: the decision at stage C depends on the token's full history, not just its current state. Two tokens both in state C may be routed differently depending on whether they have visited C before, so the Markov property fails. The fix is straightforward: replace the history-dependent rule with a memoryless one.

I therefore adopt the following routing rules, which depend only on the current state: a flagged token is always forwarded to correction; a token under correction is accepted with probability $1/2$, re-flagged with probability $1/4$, or rejected with probability $1/4$; accepted and rejected tokens are absorbing. Since each routing decision depends only on the current state, the Markov property holds by construction. The transition matrix and transition graph are shown below.



One verifies that each row of P sums to 1 (Proposition 1.3.1). If a token starts flagged ($\mu_0 = e_F$), after one step $\mu_1 = \mu_0 P = e_C$: it is under correction with certainty. After two steps $\mu_2 = \mu_0 P^2 = (1/4, 0, 1/2, 1/4)$: it has been accepted with probability $1/2$, rejected with probability $1/4$, and re-flagged with probability $1/4$. The n -step distribution is $\mu_n = \mu_0 P^n$ by Remark 1.3.2.

Remark 1.3.2. Given the initial distribution μ_0 viewed as a row vector, the distribution of X_n is $\mu_n := \mu_0 P^n$, since $(P^n)_{ij} = \mathbb{P}(X_n = j \mid X_0 = i)$ by induction on n . Since P is row-stochastic, every power P^n is row-stochastic, so μ_n is again a probability vector.

To analyse which characters can influence one another under the noise channel, and in particular whether every character in the alphabet is reachable from every other, I introduce the notion of communication between states.

Definition 1.3.4 (Communication [20]). A state j is accessible from i , written $i \rightsquigarrow j$, if $(P^n)_{ij} > 0$ for some $n \geq 1$. Two states communicate, written $i \longleftrightarrow j$, if $i \rightsquigarrow j$ and $j \rightsquigarrow i$.

Remark 1.3.3. The relation $\longleftrightarrow$ is not an equivalence relation in general: reflexivity fails for any state i with $(P^n)_{ii} = 0$ for all $n \geq 1$ (e.g. a two-state chain with $p_{12} = p_{21} = 1$, where neither state has a return path of odd length). It is however symmetric and transitive. Communicating classes are therefore defined combinatorially below.

States that mutually communicate are grouped into maximal subsets called communicating classes, whose closed or open structure determines whether the chain can escape a subset of the alphabet.

Definition 1.3.5 (Communicating class). *A communicating class is a subset $C \subseteq I$ such that:*

- (a) $i \rightsquigarrow j \quad \forall i, j \in C;$
- (b) $i \not\rightsquigarrow k \quad \forall i \in C, \forall k \notin C.$

A communicating class C is *closed* if $p_{ik} = 0$ for all $i \in C, k \notin C$: once the chain enters C , it never leaves. A state i belonging to a closed communicating class of size 1 is called *absorbing*: $p_{ii} = 1$. A Markov chain is *irreducible* if I itself is the unique communicating class, i.e. every state is accessible from every other.

Beyond accessibility, the temporal regularity of returns to a state matters for the long-run behaviour of the chain. A chain that returns to a state only at multiples of some period $d > 1$ cannot converge to a stationary distribution; aperiodicity eliminates this obstruction, and as shown below it is guaranteed for the noise channel by the presence of positive diagonal entries.

Definition 1.3.6 (Period and aperiodicity). *The period of a state $i \in I$ is*

$$d(i) := \gcd\{n \geq 1 : (P^n)_{ii} > 0\}.$$

A state is aperiodic if $d(i) = 1$. In an irreducible chain all states share the same period; the chain itself is called aperiodic if $d(i) = 1$ for any (equivalently, all) $i \in I$.

The communicating class structure and periods of Example 1.3.1 are analysed in Example 1.3.2.

Example 1.3.2 (Communicating classes and period of the pipeline). *I analyse the chain of Example 1.3.1.*

- **Communicating classes.** *States F and C communicate: $F \rightsquigarrow C$ since $p_{FC} = 1$ and $C \rightsquigarrow F$ since $p_{CF} = 1/4 > 0$, while they don't communicate with A, R (as well as A and R don't communicate between them). Their class $\{F, C\}$ is not closed, because C can reach A ($p_{CA} = 1/2$) and R ($p_{CR} = 1/4$), neither of which has any edge back into $\{F, C\}$. The classes $\{A\}$ and $\{R\}$ are each closed: $p_{AA} = p_{RR} = 1$ and no exit is possible, so both states are absorbing (Definition 1.3.5).*
- **Period.** *A token at F can only return to F via the path $F \rightarrow C \rightarrow F$, which takes exactly 2 steps; it cannot return in 1 step ($p_{FF} = 0$), nor in any odd number of steps, since every return path must traverse C and back, giving an even total length. Hence $(P^n)_{FF} > 0$ if and only if n is even, so $d(F) = \gcd\{2, 4, 6, \dots\} = 2$. By Proposition 1.3.2, $d(C) = 2$ as well. The absorbing states A and R each have a self-loop ($p_{AA} = p_{RR} = 1$), giving $d(A) = d(R) = 1$: they are aperiodic.*

The fact that the period is uniform within each communicating class, established next, justifies speaking of the period of a class rather than of a single state.

Proposition 1.3.2 (Period is a class property [20, Prop. 2.6]). *If $i \rightsquigarrow j$, then $d(i) = d(j)$.*

Proof. The case $i = j$ is trivial; assume $i \neq j$. Let $a, b \geq 1$ be integers such that $(P^b)_{ij} > 0$ and $(P^a)_{ji} > 0$. For any $n \geq 1$ with $(P^n)_{ii} > 0$,

$$(P^{a+n+b})_{jj} \geq (P^a)_{ji}(P^n)_{ii}(P^b)_{ij} > 0,$$

and analogously $(P^{a+2n+b})_{jj} \geq (P^a)_{ji}((P^n)_{ii})^2(P^b)_{ij} > 0$. Hence both $a + b + n$ and $a + b + 2n$ are multiples of $d(j)$, so their difference n is also a multiple of $d(j)$. Since this holds for every n with $(P^n)_{ii} > 0$, I conclude $d(j) \mid d(i)$. By symmetry $d(i) \mid d(j)$, hence $d(i) = d(j)$. $\square$

Remark 1.3.4. A sufficient condition for aperiodicity is the existence of a self-loop: if $p_{ii} > 0$ for some i , then $1 \in \{n \geq 1 : (P^n)_{ii} > 0\}$, so $d(i) = 1$. In an irreducible chain the period is a class property (Proposition 1.3.2), so a single self-loop forces the entire chain to be aperiodic.

To study the long-run behaviour of the chain, it is necessary to understand whether a state is revisited with certainty or only with some probability strictly less than one. This distinction, formalised through first passage times, also determines which characters act as permanent corruption targets and which are merely transient waypoints under iterated application of the noise channel.

Definition 1.3.7 (First passage time). Let $(X_n)_{n \geq 0}$ be a Markov chain on I . The first passage time to state j is the random variable $T_j : \Omega \rightarrow \mathbb{N}^* \cup \{+\infty\}$ defined by

$$T_j := \min\{n \geq 1 : X_n = j\},$$

with the convention $T_j = +\infty$ if $\{n \geq 1 : X_n = j\} = \emptyset$. When the chain starts at i and $j = i$, T_i is also called the first return time to i .

First passage times allow one to assign a probability to the event that the chain ever reaches a given state, which is the basis of the recurrence classification introduced next.

Definition 1.3.8 (First-passage probabilities). I write $\mathbb{P}_i(\cdot) := \mathbb{P}(\cdot \mid X_0 = i)$ for the law of the chain started at state i . For each pair of states $i, j \in I$ and each $n \geq 1$, the first-passage probability at step n is

$$f_{ij}^{(n)} := \mathbb{P}_i(T_j = n).$$

The total first-passage probability is

$$f_{ij}^* := \mathbb{P}_i(T_j < \infty) = \sum_{n=1}^{\infty} f_{ij}^{(n)}.$$

The quantity f_{ii}^* is the probability that the chain ever returns to i when started there.

Definition 1.3.9 (Transience and recurrence). A state $i \in I$ is transient if $f_{ii}^* < 1$ and recurrent if $f_{ii}^* = 1$. A recurrent state is positive recurrent if $\mathbb{E}_i[T_i] < \infty$ and null recurrent if $\mathbb{E}_i[T_i] = \infty$, where $\mathbb{E}_i[\cdot] := \mathbb{E}[\cdot \mid X_0 = i]$ is the expectation of the chain started at state i .

Remark 1.3.5. *Null recurrence (return is certain but takes infinitely long on average) is distinct from transience (return is not even certain). In a finite irreducible chain null recurrence cannot occur, since finiteness of I prevents return times from being infinite almost surely; see Corollary 1.3.3.*

The transience and recurrence of the states in Example 1.3.1 are identified in the following example.

Example 1.3.3 (Transience and recurrence in the pipeline). *I classify the states of Example 1.3.1 using Definition 1.3.9.*

States A and R are absorbing ($p_{AA} = p_{RR} = 1$), so $f_{AA}^ = f_{RR}^* = 1$ and $\mathbb{E}_A[T_A] = \mathbb{E}_R[T_R] = 1$: both are positive recurrent.*

States F and C are transient. Starting from F , the chain reaches either A or R with probability 1 (Proposition 1.3.3) and never returns to F thereafter, so $f_{FF}^ < 1$. To see this directly, at each visit to C the token leaves $\{F, C\}$ permanently with probability $p_{CA} + p_{CR} = 3/4$, so the probability of remaining in $\{F, C\}$ for at least k full cycles decays geometrically as $(1/4)^k \rightarrow 0$. By Corollary 1.3.1, since $F \leftrightarrow C$, state C is transient as well. The absorption probabilities and mean absorption times from F and C are computed in Example 1.3.4.*

The proofs of the classification theorems below rely on the following two classical results.

Lemma 1.3.1 (Abel's theorem). *Let $(a_n)_{n \geq 0}$ be a sequence of non-negative reals and let $A(s) = \sum_{n=0}^{\infty} a_n s^n$ have radius of convergence $r \geq 1$. Then*

$$\lim_{s \rightarrow 1^-} A(s) = \sum_{n=0}^{\infty} a_n \in [0, +\infty].$$

The renewal equation below decomposes each n -step transition probability into a sum over possible first-arrival times, providing the fundamental link between n -step probabilities and first-passage probabilities that underlies the recurrence-transience dichotomy.

Theorem 1.3.1 (Renewal equation). *For all $i, j \in I$ and all $n \geq 1$,*

$$(P^n)_{ij} = \sum_{k=1}^n f_{ij}^{(k)} (P^{n-k})_{jj}. \quad (1.3.1)$$

Proof. Since the events $\{T_j = k\}$ for $k = 1, \dots, n$ and $\{T_j > n\}$ form a partition of Ω , I decompose

$$(P^n)_{ij} = \mathbb{P}_i(X_n = j) = \sum_{k=1}^n \mathbb{P}_i(X_n = j, T_j = k) + \mathbb{P}_i(X_n = j, T_j > n).$$

The last term vanishes: if $T_j > n$ then $X_1, \dots, X_n \neq j$, so $X_n \neq j$. Hence

$$(P^n)_{ij} = \sum_{k=1}^n \mathbb{P}_i(X_n = j \mid T_j = k) \mathbb{P}_i(T_j = k).$$

On the event $\{T_j = k\}$ the chain is at j at time k , and by the Markov property the future evolution depends only on $X_k = j$. Therefore $\mathbb{P}_i(X_n = j \mid T_j = k) = (P^{n-k})_{jj}$, and recalling $f_{ij}^{(k)} := \mathbb{P}_i(T_j = k)$ gives (1.3.1). $\square$

Equipped with the renewal equation and Abel's theorem, I can now characterise recurrence and transience directly in terms of the divergence or convergence of the series $\sum_n (P^n)_{ii}$, a criterion that will later be inherited by the spectral analysis of the noise matrix.

Theorem 1.3.2 (Recurrence-transience dichotomy). *For any state $i \in I$, the following are equivalent:*

- (a) i is recurrent ($f_{ii}^* = 1$).
- (b) The series $\sum_{n=0}^{\infty} (P^n)_{ii}$ diverges.

Equivalently, i is transient if and only if the series converges, in which case

$$\sum_{n=0}^{\infty} (P^n)_{ii} = \frac{1}{1 - f_{ii}^*} < \infty. \quad (1.3.2)$$

Proof. Let $P_{ii}(s) := \sum_{n=0}^{\infty} (P^n)_{ii} s^n$ and $F_{ii}(s) := \sum_{n=1}^{\infty} f_{ii}^{(n)} s^n$, both with radius of convergence at least 1. The renewal equation (1.3.1) with $i = j$ gives $(P^n)_{ii} = \sum_{k=1}^n f_{ii}^{(k)} (P^{n-k})_{ii}$ for $n \geq 1$. Multiplying by s^n , summing over $n \geq 1$, and exchanging the order of summation (justified by non-negativity for $s \in [0, 1)$) gives $P_{ii}(s) - 1 = F_{ii}(s) P_{ii}(s)$, hence $P_{ii}(s) = 1/(1 - F_{ii}(s))$. Since $f_{ii}^* = \lim_{s \rightarrow 1^-} F_{ii}(s)$, Lemma 1.3.1 gives

$$\sum_{n=0}^{\infty} (P^n)_{ii} = \lim_{s \rightarrow 1^-} P_{ii}(s) = \frac{1}{1 - f_{ii}^*},$$

which equals $+\infty$ if and only if $f_{ii}^* = 1$, proving both equivalences simultaneously. $\square$

Since the series criterion depends only on the communicating class structure, recurrence and transience are inherited uniformly within each class, as the following two corollaries establish.

Corollary 1.3.1 (Class property). *Recurrence and transience are class properties: if $i \leftrightarrow j$, then i and j are both recurrent or both transient.*

Proof. Given i, j with $i \leftrightarrow j$, there exist $\ell, m \geq 1$ with $(P^\ell)_{ij} > 0$ and $(P^m)_{ji} > 0$. For any $n \geq 1$, $(P^{\ell+n+m})_{ii} \geq (P^\ell)_{ij} (P^n)_{jj} (P^m)_{ji}$, so $\sum_n (P^n)_{ii}$ and $\sum_n (P^n)_{jj}$ both diverge or both converge. The conclusion follows from Theorem 1.3.2. $\square$

Corollary 1.3.2 (Vanishing of transient-state probabilities). *If j is transient, then $(P^n)_{ij} \rightarrow 0$ as $n \rightarrow \infty$ for every $i \in I$.*

Proof. By (1.3.2), $\sum_{n \geq 0} (P^n)_{jj} < \infty$, so $(P^n)_{jj} \rightarrow 0$. For $i \neq j$, summing the renewal equation (1.3.1) over $n \geq 1$ and exchanging the order of summation by Fubini-Tonelli (all terms are non-negative):

$$\sum_{n=1}^{\infty} (P^n)_{ij} = \sum_{k=1}^{\infty} f_{ij}^{(k)} \sum_{m=0}^{\infty} (P^m)_{jj} = f_{ij}^* \cdot \sum_{m=0}^{\infty} (P^m)_{jj} < \infty.$$

Since the series has non-negative terms and converges, its general term tends to zero, hence $(P^n)_{ij} \rightarrow 0$. $\square$

The following lemma quantifies the rate at which $(P^n)_{ii}$ converges for a recurrent state, connecting mean return times to the limit of the diagonal entries of P^n ; it is the key ingredient in the proof of the ergodic theorem below.

Lemma 1.3.2 (Discrete renewal theorem). *Let i be a recurrent state with period $d(i)$. Then*

$$\lim_{n \rightarrow \infty} (P^{nd(i)})_{ii} = \frac{d(i)}{\mathbb{E}_i[T_i]}.$$

In particular, if $\mathbb{E}_i[T_i] = \infty$ then $(P^n)_{ii} \rightarrow 0$.

Proof. See [20, Thm. 3.4]. $\square$

In the finite irreducible case, null recurrence is ruled out entirely: finiteness of the state space forces every state to be positive recurrent, a fact that guarantees the existence of a unique stationary distribution and hence a well-defined long-run character frequency for the noise channel.

Corollary 1.3.3. *Every finite irreducible Markov chain has all states positive recurrent.*

Proof. Step 1 (existence of a recurrent state). Suppose for contradiction that every state is transient. Corollary 1.3.2 gives $(P^n)_{ij} \rightarrow 0$ for all $i, j \in I$. Since P^n is row-stochastic and I is finite,

$$1 = \lim_{n \rightarrow \infty} \sum_{j \in I} (P^n)_{ij} = \sum_{j \in I} \lim_{n \rightarrow \infty} (P^n)_{ij} = 0,$$

a contradiction.

Step 2 (all states are recurrent). By Corollary 1.3.1 and Step 1, every state in the unique communicating class I is recurrent.

Step 3 (positive recurrence). Suppose for contradiction that some state j is null recurrent. Since null recurrence is a class property (Corollary 1.3.1), every state is null recurrent. Lemma 1.3.2 gives $(P^n)_{jj} \rightarrow 0$ for every $j \in I$, and Corollary 1.3.2 gives $(P^n)_{ij} \rightarrow 0$ for $i \neq j$. The same calculation as in Step 1 yields $1 = 0$, a contradiction. $\square$

Now let $I = T \cup R_1 \cup \dots \cup R_s$ be the decomposition of the state space into the set of transient states T and the closed communicating classes $R_1, \dots, R_s$. By Corollary 1.3.2, starting from any $i \in T$ the chain must eventually enter one of the classes R_c , never to leave. I now quantify these absorption events; the concrete computation for Example 1.3.1 follows in Example 1.3.4.

For each $i \in T$ define

$$u_i^{(n)} := \mathbb{P}_i(X_1 \in T, \dots, X_n \in T), \quad u_i := \mathbb{P}_i\left(\bigcap_{n \geq 1} \{X_n \in T\}\right),$$

so u_i is the probability of remaining in T forever.

Proposition 1.3.3 (Permanence in transient states). *For every $i \in T$, $(u_i^{(n)})_{n \geq 0}$ is non-increasing with $u_i^{(n)} \rightarrow u_i$, and $(u_i)_{i \in T}$ satisfies*

$$u_i = \sum_{j \in T} p_{ij} u_j, \quad i \in T. \quad (1.3.3)$$

If T is finite, the unique solution of (1.3.3) is $u_i = 0$ for all $i \in T$.

Proof. The sequence is non-increasing because $\{X_1 \in T, \dots, X_{n+1} \in T\} \subseteq \{X_1 \in T, \dots, X_n \in T\}$. Equation (1.3.3) follows by conditioning on the first step and letting $n \rightarrow \infty$. If T is finite, iterating gives $u_i \leq \max_{j \in T} \{u_j\} \sum_{j \in T} (P^n)_{ij} \xrightarrow{n \rightarrow \infty} 0$ by Corollary 1.3.2. $\square$

Since absorption is certain for all transient states, it is meaningful to compute the probability of being absorbed into each specific closed class and the expected time until absorption. These quantities are characterised by linear systems whose unique solutions are given by the theorems below.

Fix a closed class $C \in \{R_1, \dots, R_s\}$. The *absorption time* into C is $V := \inf_{j \in C} T_j$, and the *absorption probability* from $i \in T$ is $v_i := \mathbb{P}_i(V < \infty)$.

Theorem 1.3.3 (Absorption probabilities). *The family $(v_i)_{i \in T}$ is a bounded solution of*

$$v_i = \sum_{j \in C} p_{ij} + \sum_{j \in T} p_{ij} v_j, \quad i \in T. \quad (1.3.4)$$

If T is finite, it is the unique bounded solution.

Proof. Let $v_i^{(n)} := \mathbb{P}_i(V = n)$. Clearly $v_i^{(1)} = \sum_{j \in C} p_{ij}$, and conditioning on the first step for $n \geq 2$ gives $v_i^{(n)} = \sum_{j \in T} p_{ij} v_j^{(n-1)}$. Summing over $n \geq 1$ yields (1.3.4). For uniqueness, if (v_i) and (w_i) are two bounded solutions their difference $u_i := v_i - w_i$ satisfies (1.3.3); by Proposition 1.3.3, $u_i = 0$ for all i . $\square$

Remark 1.3.6. When $s \geq 2$, applying Theorem 1.3.3 with $C = R_c$ gives $v_i^{(c)} := \mathbb{P}_i(V_{R_c} < \infty)$ for each c . Since $u_i = 0$ for all $i \in T$ (Proposition 1.3.3), absorption is certain: $\sum_{c=1}^s v_i^{(c)} = 1$ for all $i \in T$.

An analogous linear system characterises the mean time to absorption, whose finiteness is guaranteed by the same geometric decay argument used in the proof below.

Theorem 1.3.4 (Mean absorption times). *Suppose I is finite and C is the unique closed class. The mean absorption times $w_i := \mathbb{E}_i[V]$ are finite for every $i \in T$ and are the unique bounded solution of*

$$w_i = 1 + \sum_{j \in T} p_{ij} w_j, \quad i \in T. \quad (1.3.5)$$

Proof. Step 1 (finiteness). Let $u_i^{(n)} := \mathbb{P}_i(V > n)$. Since T is finite and $\sum_{j \in T} (P^m)_{ij} \rightarrow 0$ by Corollary 1.3.2, choose m large enough so that $M := \max_{i \in T} \sum_{j \in T} (P^m)_{ij} < 1$. Iterating gives $u_i^{(km)} \leq M^k$, hence

$$w_i = \sum_{n=0}^{\infty} \mathbb{P}_i(V > n) \leq m \sum_{k=0}^{\infty} M^k = \frac{m}{1-M} < \infty.$$

Step 2 (the system (1.3.5)). Conditioning on the first step: if $X_1 = j \in T$, absorption takes $1 + \mathbb{E}_j[V]$ steps in total; if $X_1 \in C$, $V = 1$. Summing over both cases gives

$$\mathbb{E}_i[V] = \underbrace{\sum_{j \in T} p_{ij} + \sum_{j \in C} p_{ij}}_{=1} + \sum_{j \in T} p_{ij} \mathbb{E}_j[V],$$

which is precisely (1.3.5).

Step 3 (uniqueness). If (w_i) and $(\tilde{w}_i)$ are two bounded solutions, their difference $d_i := w_i - \tilde{w}_i$ satisfies the homogeneous system (1.3.3). By Proposition 1.3.3, $d_i = 0$ for all $i \in T$. $\square$

Example 1.3.4 (Absorption in the pipeline). *I apply Theorems 1.3.3 and 1.3.4 to the chain of Example 1.3.1, with transient set $T = \{F, C\}$ and the two absorbing classes $\{A\}$ and $\{R\}$.*

Absorption probabilities into A . System (1.3.4) with $C = \{A\}$ gives

$$\begin{aligned} v_F &= p_{FC} v_C = v_C, \\ v_C &= p_{CA} + p_{CF} v_F = \frac{1}{2} + \frac{1}{4} v_F. \end{aligned}$$

Substituting $v_F = v_C$ into the second equation: $v_C = 1/2 + v_C/4$, so $3v_C/4 = 1/2$ and $v_C = v_F = 2/3$. By Remark 1.3.6, the probability of absorption into R is $1 - 2/3 = 1/3$ from both F and C .

Mean absorption time. System (1.3.5) gives

$$\begin{aligned} w_F &= 1 + p_{FC} w_C = 1 + w_C, \\ w_C &= 1 + p_{CF} w_F = 1 + \frac{1}{4} w_F. \end{aligned}$$

Substituting: $w_C = 1 + (1 + w_C)/4$, so $3w_C/4 = 5/4$ and $w_C = 5/3$, $w_F = 8/3$. A token starting flagged is absorbed on average after $8/3 \approx 2.67$ steps.

With absorption fully characterised, I now ask what the long-run distribution of an irreducible chain looks like. The answer is provided by the stationary distribution: a probability vector that is invariant under application of P and to which the distribution of the chain converges regardless of the initial state.

Definition 1.3.10 (Stationary distribution). *A probability vector $\pi = (\pi_j)_{j \in I}$ is a stationary distribution of the chain if*

$$\pi P = \pi, \quad \sum_{j \in I} \pi_j = 1. \quad (1.3.6)$$

If $\mu_0 = \pi$, then $\mu_n = \pi$ for all $n \geq 0$: the distribution of the chain is preserved over time.

The ergodic theorem below is the central result of this section. It guarantees that for a finite irreducible aperiodic chain the n -step distribution converges to a unique limit π independently of the starting state, which is the theoretical basis for validating the confusion matrix empirically against known English letter frequencies in Section 3.2.

Theorem 1.3.5 (Ergodic theorem [20, Thm. 3.7]). *Let $(X_n)_{n \geq 0}$ be a finite irreducible and aperiodic Markov chain on I . The following are equivalent:*

- (a) *Every state is positive recurrent.*
- (b) *There exists a stationary distribution π .*

When either condition holds, π is the unique stationary distribution, $\pi_j > 0$ for all $j \in I$,

$$\pi_j = \frac{1}{\mathbb{E}_j[T_j]}, \quad \forall j \in I, \quad (1.3.7)$$

and for every initial distribution μ_0 : $\lim_{n \rightarrow \infty} (P^n)_{ij} = \pi_j$ and $\mu_0 P^n \rightarrow \pi$ as $n \rightarrow \infty$.

Proof. Step 1 ((a) $\Rightarrow$ (b)). Fix any state $k \in I$ and let $\nu_j := \mathbb{E}_k[\sum_{n=0}^{T_k-1} \mathbf{1}_{\{X_n=j\}}]$ be the expected number of visits to j during one excursion from k . Since the chain is positive recurrent, $\sum_j \nu_j = \mathbb{E}_k[T_k] < \infty$ and a first-step argument gives $\nu P = \nu$. Setting $\pi_j := \nu_j / \mathbb{E}_k[T_k]$ yields a stationary distribution. Taking $k = j$ gives $\nu_j = \mathbb{E}_j[T_j]$, hence (1.3.7).

Step 2 ((b) $\Rightarrow$ (a)). If π exists, Step 4 gives $(P^n)_{jj} \rightarrow \pi_j > 0$, so $T_j < \infty$ almost surely and $\mathbb{E}_j[T_j] = 1/\pi_j < \infty$.

Step 3 (uniqueness). Any stationary distribution satisfies $\pi_j = 1/\mathbb{E}_j[T_j]$ by (1.3.7); since mean return times are determined by the chain, π is unique.

Step 4 (convergence). Aperiodicity gives $d(j) = 1$, so Lemma 1.3.2 yields $\lim_{n \rightarrow \infty} (P^n)_{jj} = 1/\mathbb{E}_j[T_j] = \pi_j$. For $i \neq j$, the generating-function identity $P_{ij}(s) = F_{ij}(s) P_{jj}(s)$, together with $f_{ij}^* = 1$ (irreducibility and recurrence), gives $\lim_n (P^n)_{ij} = \pi_j$. The distributional convergence $\mu_0 P^n \rightarrow \pi$ then follows by finiteness of I . $\square$

Combining the ergodic theorem with the positive-recurrence result of Corollary 1.3.3, existence and uniqueness of π follow immediately for any finite irreducible chain.

Corollary 1.3.4. *Every finite irreducible Markov chain admits a unique stationary distribution π , with $\pi_j > 0$ for all $j \in I$.*

Proof. By Corollary 1.3.3 every state is positive recurrent. Condition (b) of Theorem 1.3.5 then gives existence and uniqueness of π . $\square$

Remark 1.3.7. *For a chain with multiple closed classes, as in Example 1.3.1, the stationary distribution is not unique: any convex combination of the stationary distributions of the individual closed classes (here e_A and e_R) is itself stationary. Theorem 1.3.5 applies within each closed class separately.*

The foregoing theory applies directly to the noise model of this thesis. The confusion matrix Q estimated from real-world spelling corpora (Section 3.2) is the transition matrix of a finite Markov chain on the alphabet Σ_0 ; its ergodic properties, established above, govern the long-run distribution of corrupted characters and provide the theoretical guarantees underlying the experimental validation of Chapter 3. The next subsection makes this connection precise by introducing the character-level noise channel and its parameterisation.

Noise Model

The typo generation process is formalised as a *character-level noise channel*: each character $a \in \Sigma$ in the original string is independently replaced by a character $b \in \Sigma$ according to a row-stochastic confusion matrix Q .

Definition 1.3.11 (Character-level noise channel). *A character-level noise channel is a row-stochastic matrix $Q \in \mathbb{R}^{|\Sigma| \times |\Sigma|}$. Its entry q_{ab} denotes the probability that character a is realised as character b ; the diagonal entry q_{aa} is the probability that a is transmitted without error.*

Given an original string $w = w_1 \cdots w_\ell \in \Sigma^*$ of length $\ell := |w|$, the noisy version $\tilde{w} = \tilde{w}_1 \cdots \tilde{w}_\ell$ is obtained by drawing, for each of the ℓ positions independently,

$$\tilde{w}_k \sim Q_{w_k, \cdot}, \quad \text{i.e.} \quad \mathbb{P}(\tilde{w}_k = b \mid w_k = a) = q_{ab}, \quad k = 1, \dots, \ell.$$

The Markov-chain framework of the previous subsection applies directly to Q . To track how frequently each character appears in a corrupted string, and thereby to validate the noise model against known English letter frequencies, I introduce the occupation count.

Definition 1.3.12 (Occupation count). *Let $\tilde{w} := \tilde{w}_1 \cdots \tilde{w}_\ell$ be a corrupted string of length ℓ . The occupation count of character $a \in \Sigma$ in $\tilde{w}$ is*

$$N_\ell(a) := \sum_{k=1}^{\ell} \mathbf{1}_{\{\tilde{w}_k = a\}},$$

i.e. the number of positions in $\tilde{w}$ occupied by character a . The empirical frequency $N_\ell(a)/\ell$ is the fraction of characters in $\tilde{w}$ equal to a .

Example 1.3.5. For the corrupted string $\tilde{w} = \mathbf{abbaac}$ of length $\ell = 6$, the occupation counts are $N_6(\mathbf{a}) = 3$, $N_6(\mathbf{b}) = 2$, $N_6(\mathbf{c}) = 1$, giving empirical frequencies $1/2$, $1/3$, $1/6$ respectively. By Remark 1.3.8, if the original characters were drawn i.i.d. from a distribution μ , these frequencies would converge almost surely to $(\mu Q)_a$ as $\ell \rightarrow \infty$.

The large-sample behaviour of $N_\ell(a)/\ell$ is governed by the following standard result, stated for a generic i.i.d. sequence.

Lemma 1.3.3 (Strong law of large numbers [18, Thm. 1.10.1]). *Let $(Z_k)_{k \geq 1}$ be i.i.d. non-negative random variables with $\mathbb{E}[Z_1] = \nu \leq \infty$. Then*

$$\frac{1}{n} \sum_{k=1}^n Z_k \xrightarrow{n \rightarrow \infty} \nu \quad \text{almost surely.}$$

Proof. **Case $\nu < \infty$.** See [26, Thm. 12.1].

Case $\nu = \infty$. Fix $N < \infty$ and let $Z_k^{(N)} := Z_k \wedge N$. Since $Z_k^{(N)} \leq Z_k$, I have $n^{-1} \sum_k Z_k \geq n^{-1} \sum_k Z_k^{(N)}$. The finite- ν case gives $n^{-1} \sum_k Z_k^{(N)} \rightarrow \mathbb{E}[Z_1 \wedge N]$ almost surely. By the monotone convergence theorem, $\lim_{N \rightarrow \infty} \mathbb{E}[Z_1 \wedge N] = \infty$, so for any $M > 0$ one can choose N with $\mathbb{E}[Z_1 \wedge N] > M$, giving $\liminf_{n \rightarrow \infty} n^{-1} \sum_k Z_k > M$ almost surely. Since M was arbitrary, the sum diverges almost surely. $\square$

Remark 1.3.8. The ℓ draws $\tilde{w}_1, \dots, \tilde{w}_\ell$ are independent, but in general not identically distributed: position k has marginal $\mathbb{P}(\tilde{w}_k = a) = q_{w_k, a}$, which depends on the original character w_k at that position.

If the characters of w are themselves drawn i.i.d. from a distribution μ over Σ , every draw has the same marginal $\mathbb{P}(\tilde{w}_k = a) = (\mu Q)_a$ and the ℓ draws are i.i.d. Applying Lemma 1.3.3 to $Z_k := \mathbf{1}_{\{\tilde{w}_k = a\}}$ gives

$$\frac{N_\ell(a)}{\ell} \xrightarrow{\ell \rightarrow \infty} (\mu Q)_a \quad \text{almost surely.}$$

If moreover $\mu = \pi$ is the stationary distribution of Q , then $\mu Q = \pi$ and $N_\ell(a)/\ell \rightarrow \pi_a$.

Understanding the communicating class structure of Q allows one to characterise which characters can be reached from a given starting character under the typo model, whether the process can get trapped in a subset of the alphabet, and what the asymptotic character frequencies are in a long iteratively corrupted string (Theorem 1.3.5). In the model used in this thesis, Q is estimated from the Birkbeck and Wikipedia corpora with Laplace smoothing, ensuring $q_{ab} > 0$ for all $a, b \in \Sigma$. Every pair of characters therefore communicates in a single step and every state has a positive self-loop, so Q is irreducible and aperiodic. Corollary 1.3.3 then guarantees that every character state is positive recurrent: no character is permanently unreachable as a corruption target.

The attenuated channel introduced next interpolates between the identity and the full noise channel, providing the one-parameter family of corruption intensities used as the primary experimental axis in Chapter 4.

Definition 1.3.13 (Attenuated noise channel). *For a mutation rate $r \in [0, 1]$, the attenuated noise channel is the row-stochastic matrix*

$$M(r) := (1 - r)I + rQ,$$

where I is the identity matrix. Each character is left unchanged with probability $1 - r$ and drawn from the corruption distribution Q_a , with probability r ; setting $r = 1$ recovers the full noise channel Q , while $r = 0$ leaves every character intact.

Remark 1.3.9. *The attenuated channel $M(r)$ inherits the irreducibility and aperiodicity of Q for any $r > 0$, since $M(r)_{aa} \geq 1 - r > 0$ for all a . Its stationary distribution coincides with that of Q : $\pi M(r) = (1 - r)\pi + r\pi Q = \pi$. The family $\{M(r)\}_{r \in (0, 1]}$ thus provides a one-parameter interpolation between the identity and the full noise channel, all sharing the same long-run character distribution π .*

While the mutation rate r controls the per-character probability of corruption, the experimental design of this thesis requires a normalised notion of perturbation intensity that is comparable across queries of different lengths. This is achieved by the relative DL budget, which fixes the total edit distance of the output rather than the per-character noise level.

Definition 1.3.14 (Relative DL budget). *Let $q \in \Sigma^*$ be a query of length $\ell_q := |q|$ and let $\rho \in [0, 1]$ be the relative Damerau-Levenshtein budget. A controlled perturbation of q at level ρ is any string $q' \in \Sigma^*$ obtained by applying the character-level channel M until $\text{DL}(q, q') = \lfloor \rho \cdot \ell_q \rfloor$, where DL denotes the Damerau-Levenshtein distance (Definition 1.2.6). Equal values of ρ correspond to proportionally equal corruption intensities across queries of different lengths, enabling fair cross-query comparison.*

Remark 1.3.10. *Unlike the per-character rate r of Definition 1.3.13, which controls the probability of each individual mutation, ρ fixes the total edit distance of the output relative to the query length. This makes perturbation intensity comparable across queries of different lengths and is therefore used as the primary experimental axis in Chapter 4.*

Example 1.3.6 (Symmetric three-character noise channel). *Consider a noise model over $\Sigma = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ in which each character is kept with probability $1 - 2\varepsilon$ or replaced uniformly at random by one of the other two, each with probability $\varepsilon \in (0, 1/2)$. The transition matrix is*

$$Q = \begin{pmatrix} 1 - 2\varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 1 - 2\varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 1 - 2\varepsilon \end{pmatrix},$$

with entries $q_{aa} := 1 - 2\varepsilon$ and $q_{ab} := \varepsilon$ for $a \neq b$. Every off-diagonal entry is positive, so the chain is irreducible; every diagonal entry is positive, so every state is aperiodic. By symmetry the unique stationary distribution is uniform, $\pi = (1/3, 1/3, 1/3)$ (Corollary 1.3.4), confirming that all three characters appear with equal frequency in a long iteratively corrupted string. The attenuated channel $M(r)$ has entries $q_{aa}(r) := 1 - 2r\varepsilon$ and $q_{ab}(r) := r\varepsilon$ for $a \neq b$, preserving π while slowing mixing as r decreases.

Matrix Structure and Spectral Analysis

This subsection develops the matrix-algebraic tools underlying the spectral analysis of the noise channel. The treatment follows [27] for the Perron-Frobenius theory and the fundamental matrix, and [28] for results on spectral radius and norm convergence.

Spectral structure of primitive matrices

The long-run behaviour of a Markov chain is governed by the spectral properties of its transition matrix. I begin by recalling the algebraic setting in which these properties are established, starting from the notion of a matrix norm, which provides the topology needed to state convergence results for matrix powers.

Definition 1.3.15 (Matrix norm). *A matrix norm on $\mathbb{R}^{n \times n}$ is a norm satisfying the axioms of Definition 1.1.8 together with $\|AB\| \leq \|A\| \|B\|$ for all A, B . Standard examples are the Frobenius norm $\|A\|_F := (\sum_{i,j} a_{ij}^2)^{1/2}$ and the induced operator norm $\|A\|_{\text{op}} := \sup_{\|\mathbf{x}\|=1} \|A\mathbf{x}\|$.*

A key structural property of the noise matrix is primitivity, which strengthens irreducibility by requiring that all entries of some power A^k be strictly positive; Laplace smoothing makes this condition hold with $k = 1$.

Definition 1.3.16 (Primitive matrix). *A square matrix $A \geq 0$ is primitive if there exists $k \geq 1$ such that $A^k > 0$ entry-wise.*

Remark 1.3.11. *The smoothed confusion matrix Q is primitive by construction: Laplace smoothing makes every entry strictly positive (Definition 1.3.16 with $k = 1$). More generally, for the transition matrix P of a finite Markov chain, irreducibility together with aperiodicity implies primitivity: $(P^n)_{ij} > 0$ for all i, j and all sufficiently large n . Laplace smoothing ensures $q_{aa} > 0$ for every a , so $d(a) = 1$ and the irreducible chain is aperiodic by Proposition 1.3.2.*

Primitivity is precisely the hypothesis required by the Perron-Frobenius theorem, which characterises the dominant eigenvalue and eigenvectors of the noise matrix and thereby identifies the stationary distribution as the unique normalised positive left eigenvector.

Theorem 1.3.6 (Perron-Frobenius for primitive matrices). *Let A be an $n \times n$ non-negative primitive matrix. Then there exists an eigenvalue r such that:*

- (a) r is real and positive;
- (b) strictly positive left and right eigenvectors are associated with r ;
- (c) $r > |\lambda|$ for every eigenvalue $\lambda \neq r$;
- (d) the eigenvectors associated with r are unique up to positive scalar multiples;

- (e) if $0 \leq B \leq A$ entry-wise and β is an eigenvalue of B , then $|\beta| \leq r$;
 (f) r is a simple root of the characteristic polynomial.

Proof. See [27]. □

Remark 1.3.12. When $A = P$ is the transition matrix of a finite irreducible aperiodic chain, the Perron eigenvalue r equals 1. Row-stochasticity gives $P\mathbf{1} = \mathbf{1}$, hence 1 is a right eigenvalue and $r \geq 1$. On the other hand, for any eigenvalue λ with right eigenvector $\mathbf{x}$, $|\lambda|\|\mathbf{x}\|_\infty = \|P\mathbf{x}\|_\infty \leq \|P\|_\infty\|\mathbf{x}\|_\infty = \|\mathbf{x}\|_\infty$, so $|\lambda| \leq 1$ for every eigenvalue, giving $r = 1$. Part (c) then gives $|\lambda| < 1$ for all $\lambda \neq 1$, and part (d) identifies π as the unique normalised positive left eigenvector.

The Perron-Frobenius theorem thus identifies the stationary distribution as an eigenvector, a viewpoint that is computationally useful since π can be extracted directly from the spectrum of $P^\top$.

Corollary 1.3.5 (Stationary distribution as eigenvector). *The stationary distribution π is the unique probability vector satisfying $\pi P = \pi$, equivalently the dominant right eigenvector of $P^\top$ normalised to sum to 1.*

Proof. The equation $\pi P = \pi$ says π is a left eigenvector of P for eigenvalue 1, equivalently a right eigenvector of $P^\top$. Strict positivity and uniqueness up to scaling follow from parts (b) and (d) of Theorem 1.3.6 applied to $P^\top$; the normalisation $\mathbf{1}^\top \boldsymbol{\pi} = 1$ fixes $\boldsymbol{\pi}$ uniquely. □

Block structure and transient states

When the chain is not irreducible the state space decomposes as $I = T \cup R_1 \cup \dots \cup R_s$, and ordering states so that $T = \{1, \dots, t\}$ and $C = \{t+1, \dots, t+r\}$ for a chosen closed class C , the transition matrix takes the block form

$$P = \begin{pmatrix} P_T & R \\ 0 & B \end{pmatrix}, \quad (1.3.8)$$

where P_T is the $t \times t$ substochastic restriction to T , R is the $t \times r$ matrix of one-step jump probabilities from T to C , and B is the stochastic restriction to C .

Remark 1.3.13. When P is irreducible, as is the case for the smoothed matrix Q , the transient set is empty and (1.3.8) reduces to $P = B$. The block structure is presented here because it applies to the raw, unsmoothed noise matrix, for which characters that never appear as typo sources may form transient states or separate closed classes.

To control how fast the submatrix P_T decays to zero, I introduce the spectral radius; together with Gelfand's formula and the Neumann series, this gives invertibility of $I - P_T$ and the fundamental matrix used to compute absorption statistics in closed form.

Definition 1.3.17 (Spectral radius). *The spectral radius of $A \in \mathbb{R}^{n \times n}$ is*

$$\rho(A) := \max\{|\lambda| : \lambda \text{ is an eigenvalue of } A\}.$$

Lemma 1.3.4 (Gelfand's formula). *For any $A \in \mathbb{R}^{n \times n}$ and any submultiplicative matrix norm $\|\cdot\|$,*

$$\rho(A) = \lim_{k \rightarrow \infty} \|A^k\|^{1/k}.$$

Moreover, $\rho(A) \leq \|A^k\|^{1/k}$ for every $k \geq 1$.

Proof. See [28]. □

Gelfand's formula links the spectral radius to the norm of matrix powers, providing the equivalence between spectral and norm convergence that underlies the following proposition.

Proposition 1.3.4 (Spectral radius and norm convergence). *Let $A \in \mathbb{R}^{n \times n}$. The following are equivalent:*

- (a) $\rho(A) < 1$.
- (b) $A^n \rightarrow 0$ entry-wise as $n \rightarrow \infty$.
- (c) There exists a submultiplicative matrix norm with $\|A\| < 1$.

Proof. (a) $\Rightarrow$ (c). Lemma 1.3.4 gives $\|A^k\|_{\text{op}}^{1/k} \rightarrow \rho(A) < 1$, so choose k with $\|A^k\|_{\text{op}} < 1$. Define $\|B\|_* := \sum_{j=0}^{k-1} \|A^j B\|_{\text{op}}$; this is a submultiplicative norm with $\|A\|_* < \|I\|_*$, so normalising by $\|I\|_*$ yields a submultiplicative norm with $\|A\| < 1$.

(c) $\Rightarrow$ (b). If $\|A\| < 1$ then $\|A^n\| \leq \|A\|^n \rightarrow 0$; since all norms on $\mathbb{R}^{n \times n}$ are equivalent, $A^n \rightarrow 0$ entry-wise.

(b) $\Rightarrow$ (a). If $\rho(A) \geq 1$ there exists a unit eigenvector $\mathbf{x}$ with $\|A\mathbf{x}\| = |\lambda| \|\mathbf{x}\| \geq \|\mathbf{x}\|$, so $A^n \mathbf{x}$ does not tend to zero, contradicting (b). □

With the equivalence between spectral radius and norm convergence established, I can show that $I - P_T$ is invertible and express its inverse as a convergent series, the Neumann series, which yields the fundamental matrix directly.

Theorem 1.3.7 (Neumann series). *Let $A \in \mathbb{R}^{n \times n}$ with $\rho(A) < 1$. Then $I - A$ is invertible and $(I - A)^{-1} = \sum_{k=0}^{\infty} A^k$, where the series converges in any submultiplicative matrix norm.*

Proof. Choose a norm with $\|A\| < 1$ (Proposition 1.3.4 (c)). The partial sums are Cauchy since $\sum_{k=0}^N \|A\|^k \leq (1 - \|A\|)^{-1} < \infty$ and converge to some S . Telescoping gives $(I - A)S_N = I - A^{N+1}$; letting $N \rightarrow \infty$ and using $A^{N+1} \rightarrow 0$ yields $(I - A)S = I$. □

Corollary 1.3.6 (Spectral radius of P_T). *The substochastic matrix P_T satisfies $\rho(P_T) < 1$ and $I - P_T$ is invertible.*

Proof. By Corollary 1.3.2, $P_T^n \rightarrow 0$ entry-wise. Proposition 1.3.4 (b) $\Rightarrow$ (a) gives $\rho(P_T) < 1$; invertibility follows from Theorem 1.3.7. $\square$

The invertibility of $I - P_T$ allows one to define the fundamental matrix N , whose entries encode the expected number of visits to each transient character before the corruption process reaches a stable character class.

Definition 1.3.18 (Fundamental matrix). *The fundamental matrix of the transient part is*

$$N := (I - P_T)^{-1} = \sum_{k=0}^{\infty} P_T^k,$$

where the equality holds by Theorem 1.3.7 and Corollary 1.3.6.

Remark 1.3.14. *For the smoothed noise matrix Q , the fundamental matrix N is vacuous since $T = \emptyset$. It becomes relevant for the raw, unsmoothed matrix: N_{ij} gives the expected number of times the chain visits character j before absorption into a closed class.*

Proposition 1.3.5 (Interpretation of N). *For every $i, j \in T$,*

$$N_{ij} = \mathbb{E}_i \left[\sum_{n=0}^{\infty} \mathbf{1}_{\{X_n=j\}} \right].$$

Proof. Since $(P_T^n)_{ij} = \mathbb{P}_i(X_n = j, X_1 \in T, \dots, X_{n-1} \in T)$, summing over n gives $N_{ij} = \sum_{n=0}^{\infty} (P_T^n)_{ij} = \mathbb{E}_i[\sum_{n \geq 0} \mathbf{1}_{\{X_n=j\}}]$, where visits to $j \in T$ can only occur while the chain remains in T . $\square$

The fundamental matrix also provides a compact matrix form for the absorption equations established in Theorem 1.3.3 and Theorem 1.3.4, expressing both absorption probabilities and mean absorption times as products of N with simple vectors.

Proposition 1.3.6 (Matrix form of absorption equations). *Let $\mathbf{v} = (v_i)_{i \in T}$ be the vector of absorption probabilities into C , $\mathbf{r} := R\mathbf{1}_r$ the one-step absorption probability vector, and $\mathbf{w} = (w_i)_{i \in T}$ the mean absorption time vector. Then*

$$\mathbf{v} = N\mathbf{r}, \quad \mathbf{w} = N\mathbf{1}_t.$$

When $s \geq 2$, the absorption probability vectors for all classes $R_1, \dots, R_s$ are the columns of NR , and their row sums equal $\mathbf{1}_t$.

Proof. System (1.3.4) reads $(I - P_T)\mathbf{v} = \mathbf{r}$, giving $\mathbf{v} = N\mathbf{r}$. System (1.3.5) reads $(I - P_T)\mathbf{w} = \mathbf{1}_t$, giving $\mathbf{w} = N\mathbf{1}_t$. For $s \geq 2$, writing $R = [R^{(1)} \mid \dots \mid R^{(s)}]$ and using the row-stochastic identity $P_T\mathbf{1}_t + R\mathbf{1}_r = \mathbf{1}_t$: $\sum_{c=1}^s NR^{(c)}\mathbf{1} = NR\mathbf{1}_r = N(I - P_T)\mathbf{1}_t = \mathbf{1}_t$. $\square$

1.4 Summary

This chapter has developed the mathematical apparatus needed to study the stability of an embedding model $E : (\Sigma^*, d) \rightarrow (\mathbb{R}^d, \|\cdot\|)$ as a function between metric spaces.

- **Section 1.1** introduced the geometric framework on both sides of E . Metric spaces (Definition 1.1.1) provide the minimal structure needed for distance, convergence, and continuity, while Lipschitz and bi-Lipschitz continuity (Definition 1.1.4) refine continuity into a quantitative stability statement: an upper bound L controls displacement, a lower bound c controls discriminability, and $\kappa = C/c$ measures the distortion of E . Normed and inner product spaces equip $\mathbb{R}^d$ with the structure needed for ℓ_p distances and cosine similarity; the equivalence of norms in finite dimension (Theorem 1.1.4) guarantees that a Lipschitz bound for E exists independently of the output distance, motivating the four-distances experimental design. The curse of dimensionality and the geometry of $\mathbb{S}^{d-1}$ as a Riemannian manifold close the section, explaining why cosine similarity remains discriminative even at high d .
- **Section 1.2** developed the corresponding theory on the input side Σ^* . After formalising edit distance via a general cost-based framework (Definition 1.2.1) and identifying when it is a genuine metric (Theorem 1.2.1), the chapter examined Hamming, LCS, Levenshtein, and the transposition-aware OSA and Damerau-Levenshtein distances. The no-reuse restriction breaks the triangle inequality for OSA (Proposition 1.2.4), while lifting it restores the metric property for Damerau-Levenshtein (Theorem 1.2.2), adopted as the primary input metric of the thesis [16].
- **Section 1.3** recalled the theory of finite Markov chains underlying the character-level noise channel. Starting from the Markov property and the transition matrix (Definitions 1.3.1 and 1.3.3), the chapter built up communicating classes, periodicity, and the recurrence-transience dichotomy (Theorem 1.3.2), culminating in the ergodic theorem (Theorem 1.3.5): every finite, irreducible, aperiodic chain admits a unique stationary distribution π . The Perron-Frobenius theorem and the fundamental matrix $N = (I - P_T)^{-1}$ supplied the tools to compute absorption probabilities and mean absorption times in closed form. These results were then specialised to the confusion matrix Q , the attenuated channel $M(r) = (1 - r)I + rQ$, and the relative Damerau-Levenshtein budget ρ (Definition 1.3.14), the primary experimental axis of Chapter 4.

Together, these sections supply the metric, normed, and stochastic vocabulary needed to formalise stability of E under typographical noise.

2 Introduction to AI Systems

This chapter introduces the computational systems at the core of this thesis: embedding models, the Transformer architecture, and the Retrieval-Augmented Generation (RAG) paradigm. The treatment is oriented towards the mathematical structures underlying these systems, with particular attention to the metric spaces in which embeddings live and the role of distance functions in information retrieval.

These three components form a pipeline. An embedding model converts raw text into geometric objects: vectors in a high-dimensional space where proximity reflects semantic similarity. A Large Language Model (based on the Transformer architecture) generates fluent, contextually coherent text from an input prompt. A Retrieval-Augmented Generation system connects the two: at inference time, the user query is embedded, semantically similar documents are retrieved from an external knowledge base, and the retrieved content is supplied to the LLM as context for generating the response. The quality of the final answer depends critically on the quality of the retrieval step, which in turn depends on the stability of the embedding model under the topographical perturbations studied in this thesis.

A common architectural distinction runs through all the models discussed in this chapter. An *encoder* is a neural network that maps a variable-length input sequence to a sequence of dense contextualised representations; its purpose is to compress and contextualise the input. A *decoder* is a neural network that generates an output sequence from such representations, typically one token at a time conditioned on all previously produced tokens. The two components can be combined or used independently, giving rise to three families of models: encoder-only models, which are the basis of the embedding models studied in this thesis; decoder-only models, which are the dominant architecture for generative LLMs; and encoder-decoder models, which remain relevant for structured sequence-to-sequence tasks such as machine translation. Both embedding models and generative LLMs rely on the same token embeddings and the self-attention mechanism of the Transformer, but differ in architecture, training objective, and purpose. The chapter is structured as follows:

- **Section 2.1** develops the theory of embedding models: the tokenisation pipeline [29, 2], the geometric structure of the latent space [23, 30], the contrastive training objectives that shape it [31, 32, 33], and the formulation of embedding models as Lipschitz maps between metric spaces.

- **Section 2.2** introduces the Transformer architecture [1], the BERT family of encoder-only models [2], and the generative LLMs that power modern conversational systems [3, 34, 35].
- **Section 2.4** describes the Retrieval-Augmented Generation paradigm [4, 36], the role of the embedding model as a retrieval backbone within it, the architecture of AI agents and the ReAct loop [37, 38], and the consequences of typographical noise on retrieval quality in both standard and agentic RAG pipelines.

2.1 Embedding Models

Natural language text is composed of *words*, the linguistic units that appear in written form: `chocolate`, `running`, `unbelievable`. Computational models, however, do not operate directly on words: they process *tokens*, discrete symbols drawn from a fixed finite vocabulary $\mathcal{V}$ and produced by a *tokeniser* τ applied to the raw string. Common tokenisation strategies, such as Byte-Pair Encoding (BPE) [29] and WordPiece [2], operate at the *subword* level: they segment words into recurring fragments, so that even out-of-vocabulary strings can be represented as sequences of known pieces. For example, `unbelievable` may be tokenised as `[un, ##believ, ##able]`, and `chocolate` as `[choc, ##olate]`.

An *embedding model* is a function $E : \Sigma^* \rightarrow \mathbb{R}^d$ that maps a raw string to a fixed-dimensional vector. In practice, E is the composition of the tokeniser $\tau : \Sigma^* \rightarrow \mathcal{V}^*$ followed by a neural encoder that maps the token sequence to $\mathbb{R}^d$. Throughout this thesis, both stages are subsumed into the single notation $E(w)$ for $w \in \Sigma^*$: the Damerau-Levenshtein distance d_{DL} is always measured between raw strings, and the stability properties of E pertain to the composed function. This convention is consistent with the experimental setup of Chapter 4, where queries and documents are presented to E as strings and the embedding is retrieved as a single vector.

The tokenisation step has direct consequences for the sensitivity of E to typographical noise: a misspelling that preserves a valid subword boundary may be tokenised identically to the correct form, leaving the embedding unchanged, while a single character insertion or deletion near a boundary can yield an entirely different token sequence. Both cases are integrated into the stability analysis of Section 2.1, where E is measured as a function on Σ^* .

The latent space

The output space $\mathbb{R}^d$ is a finite-dimensional normed space, but the image of E does not fill it uniformly. In practice, the output vectors concentrate on a smooth submanifold $\mathcal{M} \hookrightarrow \mathbb{R}^d$ of much lower intrinsic dimension. This submanifold is called the *latent manifold*, and the space $\mathbb{R}^d$ in which it lives is called the *latent space*: a continuous geometric space in which distances between vectors serve as a proxy for semantic similarity between the strings that produced them [23].

The encoder-decoder structure makes the geometry of the latent space explicit: the encoder $f : \mathbb{R}^n \rightarrow \mathbb{R}^k$ compresses the input to a lower-dimensional latent code, and the decoder $g : \mathbb{R}^k \rightarrow \mathbb{R}^n$ reconstructs the original input from it. The latent manifold is then the image $\mathcal{M} := g(\mathbb{R}^k) \hookrightarrow \mathbb{R}^n$: the smooth lower-dimensional subspace onto which the data concentrate (Figure 2.1). In the embedding models studied in this thesis, the decoder is absent and only the encoder is retained; the latent manifold is determined entirely by the training objective and the architecture of E .

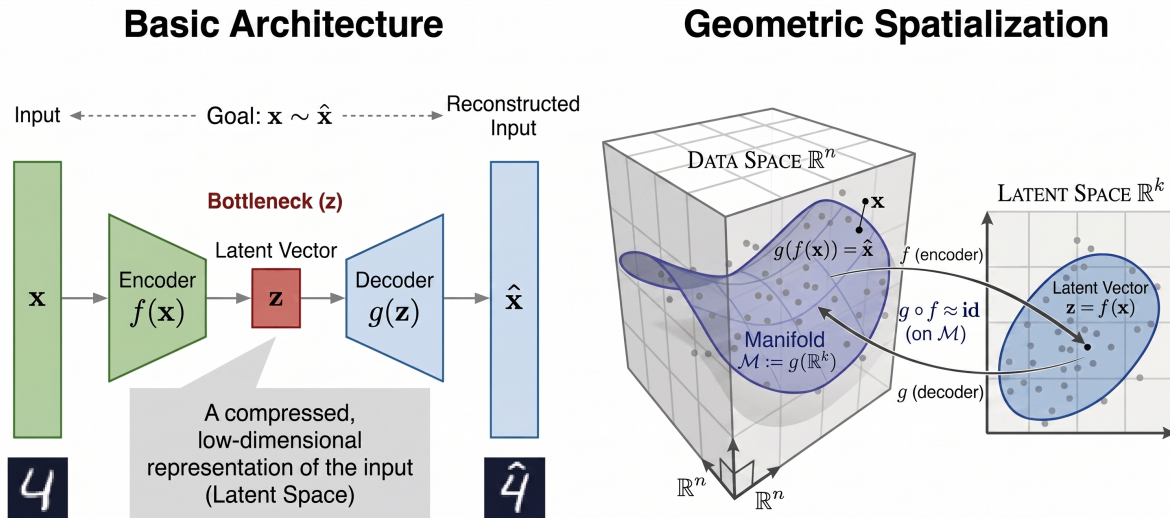


Figure 2.1. Encoder-decoder structure (left) and the resulting latent manifold in the data space (right). The encoder f compresses the input x to a latent code $z \in \mathbb{R}^k$; the decoder g reconstructs $\hat{x} = g(z) \approx x$, with $g \circ f \approx \text{id}$ on the data [68]. The latent manifold $\mathcal{M} := g(\mathbb{R}^k)$ is the smooth lower-dimensional subspace onto which the data concentrate; distances on $\mathcal{M}$ are inherited from the ambient space $\mathbb{R}^n$ [69].

The geometry of $\mathcal{M}$ is inherited from $\mathbb{R}^d$ and determines which distance function is appropriate for comparing embeddings. In the case of contrastively trained models, $\mathcal{M}$ is typically the unit sphere $\mathbb{S}^{d-1}$ (Example 1.1.3): ℓ_2 -normalisation of the output vectors is standard practice, and on $\mathbb{S}^{d-1}$ the natural distance is cosine distance (Definition 1.1.16), which measures the angle between two vectors rather than their Euclidean separation.

However, the practical consequence of operating in $d \in [768, 3072]$ dimensions is that the embedding space is subject to the curse of dimensionality [22, 23]: pairwise cosine distances concentrate near their mean value, compressing the dynamic range available to nearest-neighbour retrieval and motivating the use of multiple distance functions in the experimental evaluation.

From words to dense vectors

The simplest approach to representing words as vectors is *one-hot encoding*: given a vocabulary $\mathcal{V}$ of size $|\mathcal{V}|$, each word is mapped to a vector in $\mathbb{R}^{|\mathcal{V}|}$ with a single coordinate equal to 1 and all others equal to 0. In the *bag-of-words* model, a document

is represented as the sum of the one-hot vectors of its words. This representation is high-dimensional, sparse, and entirely uninformative about meaning: every pair of distinct words is orthogonal by construction, so **love** and **like** are geometrically as far apart as **love** and **table**.

The theoretical foundation for a more meaningful vector representation was laid by Zellig Harris in 1954. In *Distributional Structure* [30], Harris proposed that the meaning of a linguistic element is characterised not by its intrinsic properties but by the set of environments in which it can occur. The *distributional hypothesis* states that elements with similar distributions tend to have similar meanings: difference of meaning correlates with difference of distribution. For a word such as **bank**, its meaning is not deducible from the string of characters alone; it is recoverable from the words that surround it (“*deposit, overdraft, loan*” versus “*river, flood, current*”). The distributional hypothesis converts this observation into a design principle: a good representation of a word should reflect the statistical regularities of its contexts across a large corpus.

This is precisely the quantity that Word2Vec [39] and GloVe [40] optimise sixty years later. Word2Vec [39] replaces one-hot encoding with dense, low-dimensional vectors. It is a shallow two-layer neural network trained on one of two objectives. In the *Continuous Bag-of-Words* (CBOW) architecture, the context words surrounding a masked position are used to predict the missing word: given “*The _ was chocolate flavoured*”, the model is trained to output “*cake*”. The *Skip-gram* architecture reverses the task: given a target word, the model predicts the words that surround it. In both cases, the learned weight matrix becomes the embedding: each word in $\mathcal{V}$ is assigned a dense vector in $\mathbb{R}^d$ for a chosen dimension $d \ll |\mathcal{V}|$.

It is important to distinguish two different senses in which the word *context* appears in this setting. During training, context serves as the supervisory signal: the model is optimised to predict surrounding words, so co-occurrence statistics across the entire corpus are implicitly encoded into the weights. The resulting vectors are not, however, context-dependent in any dynamic sense. Once training is complete, each word in $\mathcal{V}$ is assigned a single fixed vector, used identically in every sentence. The word **bank** receives the same embedding regardless of whether it appears in a financial or a geographical context; disambiguation by context is impossible within this framework. Context thus shapes the vectors *during learning*, but plays no role *at inference time*.

The result is nonetheless a latent space with a rich geometric structure. Directions in the space acquire semantic meaning: words sharing a relationship are connected by approximately parallel displacement vectors, as shown in Figure 2.2.

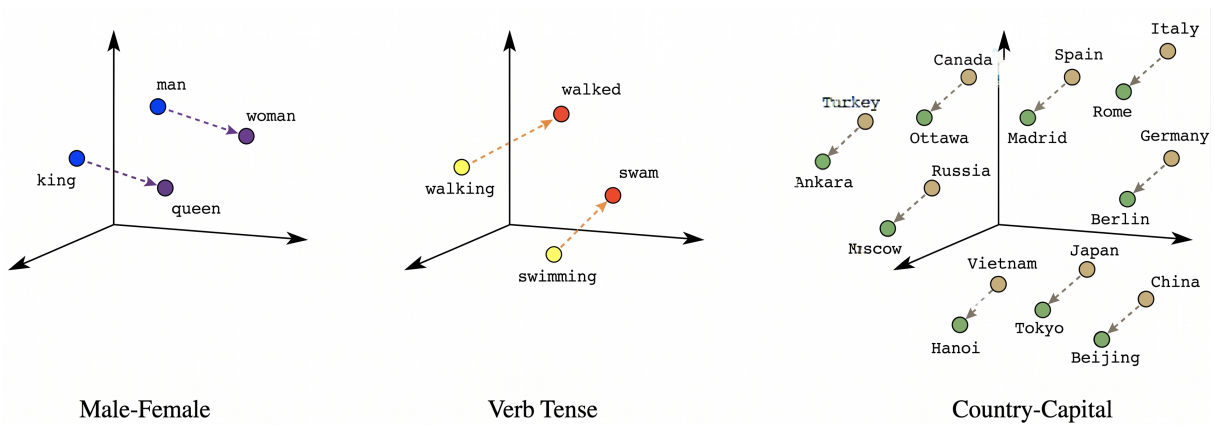


Figure 2.2. Linear structure of the Word2Vec latent space [39]. In each panel, semantically or syntactically related pairs are connected by approximately parallel vectors, reflecting a consistent linear offset in $\mathbb{R}^d$. *Left*: the gender offset $E(\text{man}) - E(\text{woman})$ is parallel across the pair *king* and *queen*. *Centre*: the tense offset $E(\text{walked}) - E(\text{walking}) \approx E(\text{swam}) - E(\text{swimming})$ is shared across verb pairs. *Right*: the capital-country offset $E(\text{Rome}) - E(\text{Italy})$ is approximately constant across country-capital pairs. Image: [70].

A classic illustration is the linear offset

$$E(\text{king}) - E(\text{man}) + E(\text{woman}) \approx E(\text{queen}),$$

where the vector $E(\text{king}) - E(\text{man})$ isolates the concept of royalty from gender, and adding $E(\text{woman})$ recovers the female-royalty direction [39]. In practice, the approximation is imperfect: the true embedding of **queen** tends to lie somewhat farther from the predicted point than this illustration suggests, presumably because the word carries usage patterns (political, musical, cultural) that go beyond a simple feminine analogue of **king**. The phenomenon generalises beyond gender: in some models, subtracting the Germany embedding from the Japan embedding and adding the embedding of *sushi* recovers a point very close to *bratwurst*, suggesting that the model has learned to associate specific directions in the space with national cuisines.

More formally, the *dot product* $\langle u, v \rangle$ between two embedding vectors serves as a measure of directional alignment: it is positive when the vectors point in similar directions, near zero when they are orthogonal, and negative when they are opposed. One consequence is that the dot product of a hypothesised direction in the embedding space (such as $E(\text{cats}) - E(\text{cat})$ as a plurality direction) against other word embeddings allows one to measure quantitatively how strongly each word aligns with that direction. This geometric intuition reappears as the core operation of scaled dot-product attention (Section 2.2), where it determines how much each token should attend to every other. The experimental evaluation of this thesis operates instead on the trained output vectors, measuring stability under multiple distance functions at inference time; this is discussed in Section 2.1 and Remark 2.1.1.

From static to contextualised representations. The two limitations of Word2Vec and GloVe noted above static embeddings and word-level operation motivate the

Transformer-based models studied in this thesis. The first limitation is the more fundamental: a single fixed vector per word cannot capture the context-dependence that is often decisive for meaning. The word `model` carries a different sense in “*a machine learning model*” and “*a fashion model*”; Word2Vec is structurally incapable of representing this distinction, since it commits to one vector at training time. The second limitation is that out-of-vocabulary strings are unrepresentable.

Modern encoder-based models overcome both limitations. They operate at the subword level via the tokeniser τ , and they produce representations that depend on the full context of the input string. Each token $t_i \in \mathcal{V}$ is mapped to a learnable vector $e_i \in \mathbb{R}^d$ via a token embedding matrix $W_E \in \mathbb{R}^{|\mathcal{V}| \times d}$, which performs the same one-hot-to-dense projection as Word2Vec but is learned end-to-end as part of the full model. Since the neural encoder processes all tokens simultaneously, positional encodings $PE_i \in \mathbb{R}^d$ are added to each token vector to inject order information (Section 2.2). The resulting input $(e_1 + PE_1, \dots, e_m + PE_m)$ is processed by a stack of Transformer encoder layers. Crucially, these layers allow every token vector to attend to every other token in the sequence: the representation of `bank` is progressively updated by the surrounding words, so that a financial context and a geographical context yield different final vectors for the same surface form. The result is a set of contextualised token representations $(h_1, \dots, h_m) \in (\mathbb{R}^d)^m$. To obtain a single vector for the full string, a *pooling* operation is applied, with two possible strategies:

- **CLS-token pooling:** a special token [CLS] (from *classification*) is prepended to the input before encoding. As it passes through the Transformer layers, it attends to every other token in the sequence; by the final layer, its representation h_1 has accumulated contextual information from the entire input and is used as the sequence embedding [2]. This strategy is most effective when the model has been fine-tuned for the target task; without fine-tuning, the [CLS] vector may not be more informative than any other token representation.
- **Mean pooling:** the embedding is the arithmetic mean $\bar{h} := \frac{1}{m} \sum_{i=1}^m h_i$ over all token representations. This distributes importance uniformly across the sequence and tends to produce more robust embeddings for general-purpose semantic similarity, at the cost of potentially averaging out task-specific signals; it is the standard choice in sentence embedding models [33].

The two strategies involve a trade-off: CLS-token pooling is well suited to classification tasks when the model is fine-tuned, while mean pooling is more reliable for general-purpose retrieval and semantic similarity across varying sequence lengths. The final vector $E(w) \in \mathbb{R}^d$ is typically ℓ_2 -normalised, projecting it onto $\mathbb{S}^{d-1}$.

The full pipeline (tokenisation, positional encoding, contextual encoding, pooling, and normalisation) is illustrated in Figure 2.3.

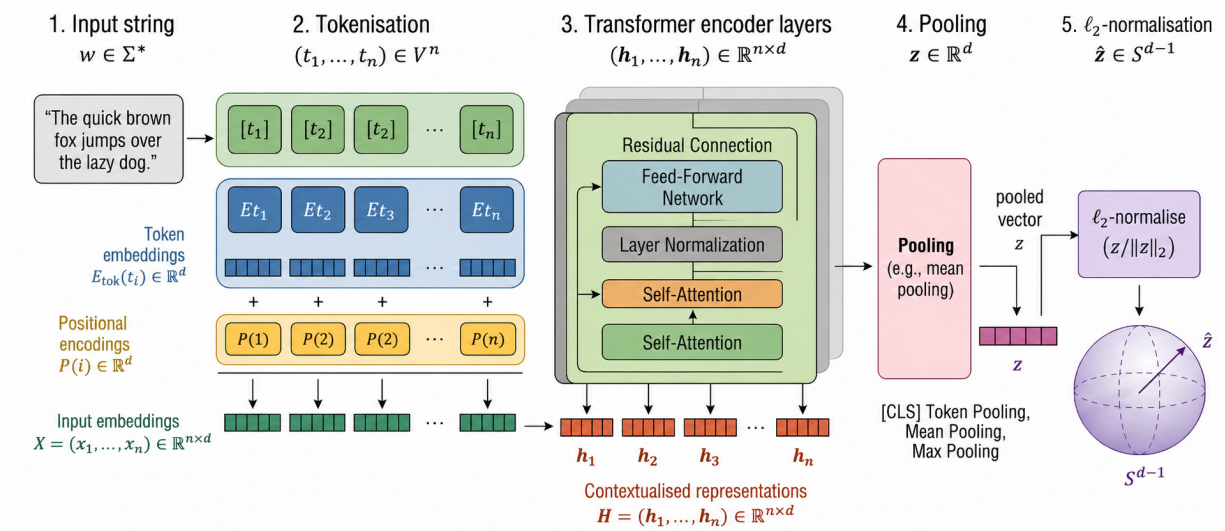


Figure 2.3. Pipeline of an encoder-based embedding model. The input string $w \in \Sigma^*$ is tokenised into a sequence $(t_1, \dots, t_m) \in \mathcal{V}^m$; positional encodings are added to the token embeddings to inject word-order information; a stack of Transformer encoder layers produces contextualised representations $(h_1, \dots, h_m)$; a pooling operation reduces these to a single vector, which is ℓ_2 -normalised onto $\mathbb{S}^{d-1}$.

Training with contrastive objectives

The quality of the embedding space is determined by the training objective. The geometry of the latent manifold is not fixed by architecture alone; it is shaped by the loss function used during training. The dominant paradigm is *metric learning* via contrastive loss [31]: the model is trained on triplets (w, w^+, w^-) , where w^+ is semantically related to w and w^- is not. The *triplet loss* is

$$\mathcal{L}(w, w^+, w^-) := \max(0, d(E(w), E(w^+)) - d(E(w), E(w^-)) + \alpha),$$

where d is a distance on $\mathbb{R}^d$ and $\alpha > 0$ is a margin. Minimising this loss pulls positive pairs together and pushes negative pairs apart, with a separation of at least α .

A widely used variant is the *InfoNCE loss* [32], also known as multiple negatives ranking loss. Given a batch of B positive pairs $\{(w_i, w_i^+)\}_{i=1}^B$, the loss for anchor w_i is a softmax cross-entropy over cosine similarities:

$$\mathcal{L}_i := -\log \frac{\exp(\cos(E(w_i), E(w_i^+))/\tau)}{\sum_{j=1}^B \exp(\cos(E(w_i), E(w_j^+))/\tau)},$$

where $\tau > 0$ is a temperature parameter controlling the sharpness of the distribution. The remaining $B - 1$ positive examples in the batch serve as implicit negatives, making large-batch training particularly effective. This objective is the standard training loss for state-of-the-art embedding models such as Sentence-BERT [33], E5 [41], and GTE [42].

The geometric consequences of this training procedure are the subject of the next subsection, which recasts E as a map between metric spaces and makes precise what it means for such a map to be stable.

Embedding models as metric maps

The preceding subsections have described the architecture and training of embedding models from an engineering perspective. We now re-read them through the mathematical lens developed in Chapter 1, considering E as a function between metric spaces and identifying the quantities that characterise its stability. This is the subsection where the two strands of the thesis, the mathematical background and the computational systems, converge into the central research question.

From a mathematical standpoint, E is a function between metric spaces: the discrete input space $(\Sigma^*, d_{\text{DL}})$, equipped with the Damerau-Levenshtein distance introduced in Section 1.2, and the normed output space $(\mathbb{R}^d, \|\cdot\|_2)$. As shown in Proposition 1.2.2, $(\Sigma^*, d_{\text{DL}})$ is a discrete metric space, so every function defined on it is trivially continuous; Lipschitz-type conditions are therefore the natural tool for studying stability. The Lipschitz constant of E is the smallest $L \geq 0$ such that

$$\|E(w_1) - E(w_2)\|_2 \leq L d_{\text{DL}}(w_1, w_2) \quad \forall w_1, w_2 \in \Sigma^*. \quad (2.1.1)$$

If E is Lipschitz, perturbations in the input are controlled in the output: a bounded edit distance implies bounded variation of the embedding vector. However, Lipschitz continuity alone does not prevent distinct inputs from being mapped to nearby vectors; it provides an upper bound on displacement but says nothing about a lower bound. This leads to the notion of *bi-Lipschitz* behaviour, which requires distances to be preserved up to controlled distortion in *both* directions. Formally, it requires the existence of constants $0 < c \leq C < \infty$ such that

$$c \cdot d_{\text{DL}}(w_1, w_2) \leq \|E(w_1) - E(w_2)\|_2 \leq C \cdot d_{\text{DL}}(w_1, w_2) \quad \forall w_1, w_2 \in \Sigma^*, \quad (2.1.2)$$

as in Definition 1.1.4. The upper constant C controls *stability*: small edits produce bounded displacement in the latent space. The lower constant c controls *discriminability*: distinct strings remain separated. Their ratio $\kappa := C/c$ is the *distortion* of the embedding; from this perspective, E acts as a *relaxed isometry* whose quality is measured by how close κ is to 1. When the lower bound fails, the embedding exhibits *metric collapse*: semantically distinct strings are mapped to nearly identical vectors and the retrieval system loses the ability to discriminate between them. This effect is illustrated in Figure 2.4: queries at small DL distance from the reference (T2, T3) land in a neighbourhood of its embedding, while semantically shifted variants (T4, T5) drift to distant regions of the latent manifold.

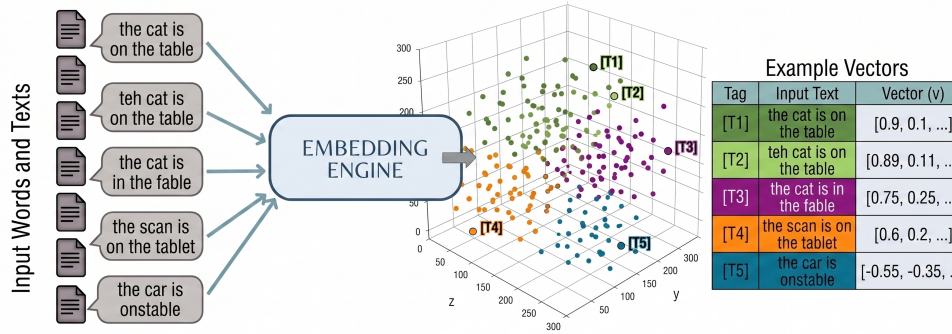


Figure 2.4. The embedding model $E : (\Sigma^*, d_{\text{DL}}) \rightarrow (\mathbb{R}^d, \|\cdot\|_2)$, with output visualised in $\mathbb{R}^3$ via dimensionality reduction. [T1] is the reference query; [T2] differs by a single transposition (*teh*, $d_{\text{DL}} = 1$) and maps close to [T1]; [T3] involves a word substitution (*table*→*fable*, $d_{\text{DL}} = 2$) and drifts slightly further; [T4] and [T5] introduce semantic shifts and land in substantially different regions.

A more granular stability measure than the global $\text{Lip}(E)$ is the *local Lipschitz constant* at scale ρ , defined by restricting attention to pairs of strings whose relative DL distance lies within a controlled budget:

$$\text{Lip}_\rho(E) := \sup \left\{ \frac{\|E(w_1) - E(w_2)\|_2}{d_{\text{DL}}(w_1, w_2)} : 0 < \frac{d_{\text{DL}}(w_1, w_2)}{|w_1|} \leq \rho \right\}.$$

As ρ increases, more distant pairs are included and $\text{Lip}_\rho(E)$ is non-decreasing; the curve $\rho \mapsto \text{Lip}_\rho(E)$ therefore captures how stability degrades as noise intensity grows. In the empirical analysis of Chapter 4, $\text{Lip}_\rho(E)$ is estimated by the empirical upper Lipschitz constant $C(\rho)$ of the NED family (Section 3.3), evaluated across the relative DL budget range $\rho \in [0.01, 0.25]$ and over four output-space distance functions, for each of the embedding models in Table 2.1.

Remark 2.1.1 (Geometry of the trained embedding space). *Contrastive training with cosine similarity implicitly constrains output vectors to the unit sphere $\mathbb{S}^{d-1}$: since $\cos(u, v)$ depends only on the directions of u and v , the effective degrees of freedom are the angular coordinates. On $\mathbb{S}^{d-1}$, cosine and Euclidean distance are related by $\|u - v\|_2^2 = 2d_{\cos}(u, v)$ (Definition 1.1.16), so nearest-neighbour retrieval under cosine similarity is equivalent to nearest-neighbour retrieval under Euclidean distance after ℓ_2 -normalisation. It should be noted that the ℓ_p distances used in the experimental evaluation of Chapter 4 are not part of the training objective, which employs cosine similarity exclusively via the InfoNCE loss (Section 2.1). The four output-space distances (cosine, Euclidean, Manhattan, and Chebyshev) are instead used at evaluation time to measure the stability of the trained embedding function under typographical perturbations. They are topologically equivalent on $\mathbb{R}^d$ (Theorem 1.1.4) and yield qualitatively concordant stability profiles, but differ in how they weight individual coordinate deviations, providing complementary diagnostic information about the geometry of the embedding displacement.*

2.2 The Transformer

The Transformer [1], introduced in 2017, is the architectural foundation of all models discussed in this thesis. It departs from earlier recurrent architectures by processing the entire input sequence in parallel through a self-attention mechanism: rather than reading tokens one at a time and propagating information through a hidden state, every position in the sequence attends to every other position simultaneously, enabling the capture of long-range dependencies and large-scale distributed training.

One of the main problems of recurrent architectures is that they are susceptible to the *vanishing gradient problem*. During backpropagation, the gradient of the loss $\mathcal{L}$ with respect to a weight W at layer ℓ is computed via the chain rule as a product of Jacobians over all subsequent layers,

$$\frac{\partial \mathcal{L}}{\partial W^{(\ell)}} = \frac{\partial \mathcal{L}}{\partial h^{(L)}} \prod_{k=\ell}^{L-1} \frac{\partial h^{(k+1)}}{\partial h^{(k)}},$$

where each factor $\partial h^{(k+1)} / \partial h^{(k)}$ involves the derivative of the activation function. For saturating activations such as tanh or sigmoid, these derivatives lie in $(0, 1)$, so their product over $L - \ell$ layers decays exponentially. As a consequence, gradients reaching the earliest layers are negligibly small and those layers effectively stop learning. In recurrent networks this is compounded by the depth in the time dimension: a sequence of length T requires backpropagation through T time steps, each contributing a multiplicative factor.

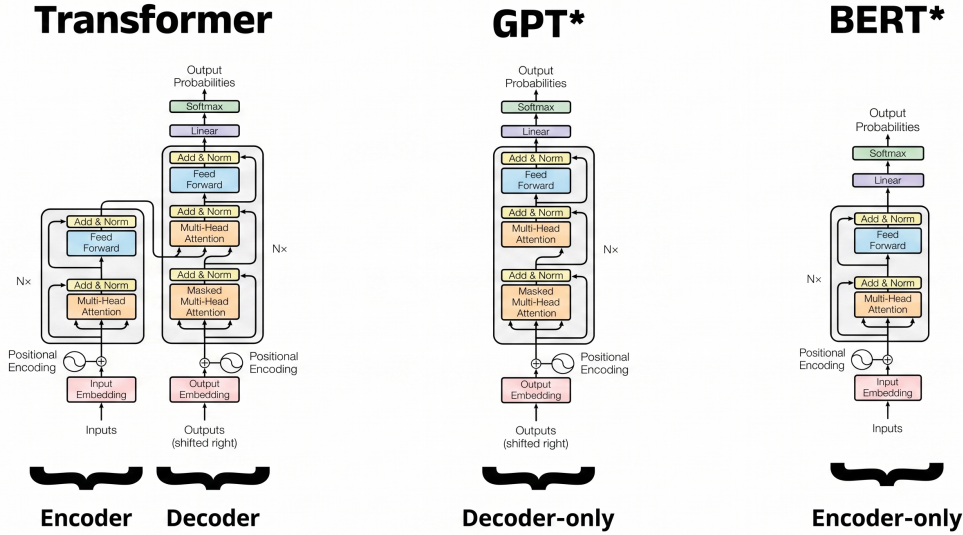
The Transformer avoids this problem entirely through its attention mechanism: since every token attends directly to every other token in a single computation, the gradient path between any two positions has constant depth regardless of sequence length. This architectural choice is the primary reason the Transformer superseded recurrent models and enabled training at the scales observed in modern LLMs.

The Transformer architecture

The Transformer [1] was originally designed as an encoder-decoder pair for machine translation, but the two components can be used independently. Depending on which is retained, three families of models arise, illustrated in Figure 2.5:

- **Encoder-only** (e.g. BERT [2]): retains the bidirectional encoder stack, producing contextualised representations of the full input. Used for understanding and representation tasks; the architectural basis of the embedding models in this thesis (Section 2.2).
- **Decoder-only** (e.g. GPT [3], LLaMA, Gemini): retains the causal decoder stack, generating output autoregressively via masked self-attention. The dominant architecture for generative LLMs since GPT-3, owing to its scalability and strong in-context learning performance (Section 2.3).
- **Encoder-decoder** (e.g. T5 [43], BART): combines both components, treating every

task as a sequence-to-sequence problem. Remains competitive for structured tasks such as machine translation and summarisation.



*Illustrative example, exact model architecture may vary slightly

Figure 2.5. The three Transformer families. *Left:* the original encoder-decoder architecture, in which the encoder (bidirectional self-attention) processes the input and the decoder (masked self-attention followed by cross-attention) generates the output token by token [1]. *Centre:* the decoder-only design (e.g. GPT [3]), generating text autoregressively. *Right:* the encoder-only design (e.g. BERT [2]), producing contextualised representations without generating text. Image: [71].

We now describe the principal building blocks shared by all three families, following the original architecture of [1].

Building blocks. Both encoder and decoder are composed of a stack of N identical layers (in the original Transformer, $N = 6$; in GPT-3, $N = 96$). Each layer contains two sub-layers: a multi-head self-attention mechanism and a position-wise feed-forward network. Both sub-layers are wrapped with a residual connection and a layer normalisation:

$$\text{output} = \text{LayerNorm}(x + \text{Sublayer}(x)).$$

Residual connections allow gradients to flow directly through the network without passing through non-linear activations at every step, alleviating the vanishing gradient problem discussed above.

Positional encoding. The self-attention mechanism is permutation-equivariant: it produces the same output regardless of the order in which tokens are presented. To

inject word-order information, each input token x_t is augmented with a positional encoding before being processed:

$$h_t^{(0)} := \text{Emb}(x_t) + \text{PE}(t).$$

The original Transformer uses sinusoidal encodings [1]:

$$\text{PE}_{(t, 2k)} := \sin\left(\frac{t}{10000^{2k/d}}\right), \quad \text{PE}_{(t, 2k+1)} := \cos\left(\frac{t}{10000^{2k/d}}\right),$$

where t is the token position and k indexes the embedding dimension. This choice allows the model to handle sequences of arbitrary length without fixing a maximum position at training time. Modern variants replace sinusoidal encodings with *learned positional embeddings* (as in BERT [2]) or with *rotary position embeddings* (RoPE [44]), which encode relative positions directly into the attention computation and scale better to long contexts.

Scaled dot-product attention. The core computational primitive is *scaled dot-product attention*. Given an input sequence of T tokens, each represented as a vector of dimension d , three matrices are obtained by separate learned linear projections:

- $Q \in \mathbb{R}^{T \times d_k}$, the **query** matrix: each of the T rows encodes what the corresponding token is looking for in the sequence. d_k is the dimensionality of the query and key projections.
- $K \in \mathbb{R}^{T \times d_k}$, the **key** matrix: each row encodes what the corresponding token offers for matching against other tokens' queries.
- $V \in \mathbb{R}^{T \times d_v}$, the **value** matrix: each row carries the actual content that the corresponding token will contribute to the output if selected. d_v is the dimensionality of the value projection, and may differ from d_k .

The attention output is computed as:

$$\text{Attention}(Q, K, V) := \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V,$$

where $\text{softmax}(\mathbf{z})_i := e^{z_i} / \sum_j e^{z_j}$ converts raw similarity scores into a probability distribution over positions. The dot product $QK^\top$ measures the compatibility between each query and each key: it is large and positive when a query vector aligns with a key vector, near zero when they are orthogonal, and negative when they point in opposing directions. This is exactly the dot-product similarity measure introduced in Section 2.1: by using it here, the model can identify which positions are most relevant to each other and weight their values accordingly. The softmax normalises these scores into attention weights, and the result is a weighted sum of the value vectors. The scaling factor $1/\sqrt{d_k}$ prevents the dot products from growing large in magnitude as dimensionality increases, which would push the softmax into regions of near-zero gradient and destabilise training.

Example 2.2.1. Consider a three-token sequence: river, bank, loan. The word bank is semantically ambiguous it can refer to a financial institution or to the side of a river and the attention mechanism should use the surrounding context to produce a contextualised representation that reflects which sense is more active in the current sentence.

Suppose the learned projections encode two latent dimensions ($d_k = d_v = 2$): the first captures natural/environmental content; the second captures financial content. Under this interpretation, the three tokens receive the following query and key projections:

- River activates only the nature dimension: $[1, 0]$.
- Loan activates only the financial dimension: $[0, 1]$.
- Bank, being ambiguous, activates both: $[1, 1]$.

In this simplified example $Q = K$: both projections are driven by the same two-dimensional semantic signal of each token. In practice, Q and K are obtained via separate learned weight matrices ($W^Q \neq W^K$), so what a token looks for (query) and what it advertises (key) can differ; the simplification keeps the illustration transparent without loss of generality.

The value matrix V represents the semantic content each token contributes to others once selected by the attention weights. It is projected separately from Q and K and encodes the raw content payload of each token before any contextualisation:

- River: $V_1 = [1, 0]$, pure nature content.
- Bank: $V_2 = [1, 1]$, both nature and financial content, reflecting its inherent ambiguity; it is up to the attention weights to determine which component is absorbed by each recipient.
- Loan: $V_3 = [0, 1]$, pure financial content.

This gives:

$$Q = K = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad V = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

Step 1: raw similarity scores. The matrix $QK^\top$ has entry (i, j) equal to the dot product of the i -th query row with the j -th key row, measuring how much token i should attend to token j :

$$QK^\top = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

Token river (row 1) finds river and bank equally relevant (score 1 each), because its nature-only query $[1, 0]$ matches the first component of both their keys; loan scores 0 because its key $[0, 1]$ is orthogonal to river's query. Token bank (row 2) attends most strongly to itself (score 2, the only token whose key $[1, 1]$ aligns on both dimensions

simultaneously) and equally to its two neighbours (score 1 each). Token loan (row 3) mirrors river symmetrically: it finds bank and itself relevant and river irrelevant.

Step 2: scaling and softmax. Dividing by $\sqrt{d_k} = \sqrt{2}$ and applying row-wise softmax converts scores into attention weights that sum to 1:

$$A := \text{softmax}\left(\frac{QK^\top}{\sqrt{2}}\right) \approx \begin{pmatrix} 0.40 & 0.40 & 0.20 \\ 0.25 & 0.50 & 0.25 \\ 0.20 & 0.40 & 0.40 \end{pmatrix}.$$

Token bank assigns half its attention weight to itself and splits the remainder equally between its two neighbours. The asymmetry between river and loan is also visible: river concentrates weight on the nature side of the sequence (higher weight on itself and bank), while loan mirrors this toward the financial side.

Step 3: weighted aggregation. The new representation of each token is computed as the weighted sum of all value rows, using the attention weights from A . Carrying out the multiplication explicitly for each token:

$$\begin{aligned} \text{river:} \quad & 0.40 \cdot [1, 0] + 0.40 \cdot [1, 1] + 0.20 \cdot [0, 1] = [0.80, 0.60], \\ \text{bank:} \quad & 0.25 \cdot [1, 0] + 0.50 \cdot [1, 1] + 0.25 \cdot [0, 1] = [0.75, 0.75], \\ \text{loan:} \quad & 0.20 \cdot [1, 0] + 0.40 \cdot [1, 1] + 0.40 \cdot [0, 1] = [0.60, 0.80]. \end{aligned}$$

The outputs are interpretable. Token river emerges as $[0.80, 0.60]$: the nature component dominates, because river attends primarily to itself and to bank's ambiguous content, drawing little from the financial side. Token loan emerges as $[0.60, 0.80]$: the financial component dominates symmetrically. Token bank lands exactly at $[0.75, 0.75]$: it remains perfectly balanced, because the nature pull from river and the financial pull from loan are equal and opposite. Had the sentence been interest, bank, loan instead, both neighbours would project a financial key, bank would assign most of its attention weight to financially-aligned tokens, and its output would shift toward a predominantly monetary representation. This is precisely what contextualisation means: the same input token produces a different output representation depending on what surrounds it, and the attention weights are the mechanism by which that surrounding context is selectively incorporated.

Note that the three output vectors above constitute the complete output of a single attention head, that is, one application of $\text{Attention}(Q, K, V) = AV$. In multi-head attention, described in the next paragraph, h such operations run in parallel over different projections of the same input, and their outputs are concatenated before being linearly combined. Each head can therefore specialise in a different type of dependency, with no single head required to capture all of them at once.

Multi-head attention. Rather than computing a single attention function over the full d -dimensional representation, the Transformer runs h attention operations in parallel, each over a different linearly projected subspace of Q , K , and V , then concatenates their outputs:

$$\text{MultiHead}(Q, K, V) := \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O,$$

$$\text{head}_i := \text{Attention}(QW_i^Q, KW_i^K, VW_i^V),$$

where $W_i^Q \in \mathbb{R}^{d \times d_k}$, $W_i^K \in \mathbb{R}^{d \times d_k}$, and $W_i^V \in \mathbb{R}^{d \times d_v}$ are the per-head projection matrices; $W^O \in \mathbb{R}^{hd_v \times d}$ is a final linear projection that combines the concatenated outputs back into the model dimension d . Running multiple heads in parallel allows the model to simultaneously capture different types of dependencies (syntactic, semantic, positional), each in its own representational subspace, without any single head needing to represent all of them at once. In the original paper [1], $h = 8$ heads were used with $d_k = d_v = d_{\text{model}}/h = 64$.

Feed-forward layers. Each Transformer layer contains, in addition to the attention sub-layer, a position-wise *feed-forward network* (FFN):

$$\text{FFN}(x) := \max(0, xW_1 + b_1) W_2 + b_2,$$

where W_1 and W_2 are learned weight matrices and b_1, b_2 are learned bias vectors; the $\max(0, \cdot)$ non-linearity (ReLU) is applied element-wise. Unlike the attention sub-layer, the FFN operates on each position independently and in parallel, with no information exchange between positions. While the attention sub-layer determines *which* tokens are relevant to each other, the FFN processes each token representation by pattern-matching it against a large collection of learned features and updating the representation accordingly. The FFN typically expands the representation to a hidden dimension of $4d$ before projecting back, making it the dominant contributor to the total parameter count in large models.

Encoder vs. decoder. The two components differ in the *direction* of attention. The **encoder** applies attention without any masking: every token x_t can attend to every other token in the sequence, in both directions, producing *bidirectional* contextualised representations. The **decoder** introduces a causal mask that sets to $-\infty$ all entries corresponding to future positions before the softmax, so that position t can only attend to $x_1, \dots, x_{t-1}$. This *unidirectional* (or causal) design preserves the autoregressive property: the model generates one token at a time, conditioning only on what has already been produced. In **encoder-decoder** models, the decoder also includes a *cross-attention* sub-layer that attends to the full encoder output, integrating the encoded input at each generation step.

The distinction between bidirectional and unidirectional attention is a property of the specific component, not of the Transformer architecture as a whole, and it has direct consequences for how the two families of models are trained and used, as developed in the next sections.

BERT and encoder-only models

BERT (Bidirectional Encoder Representations from Transformers) [2] is the encoder-only architecture that directly underpins the embedding models evaluated in this thesis. Retaining only the encoder stack means that every token attends to every other token

in both directions. As illustrated in Figure 2.6, a bidirectional encoder processing “A man was [MASK] on a river bank” exploits both left context (“A man was”) and right context (“on a river bank”) simultaneously to recover the masked word; a unidirectional model sees only the left context and may fail to disambiguate. BERT is therefore not a generative LLM under the definition adopted in Section 2.3; it is a contextualised encoder whose output is a sequence of dense vector representations rather than a distribution over tokens.

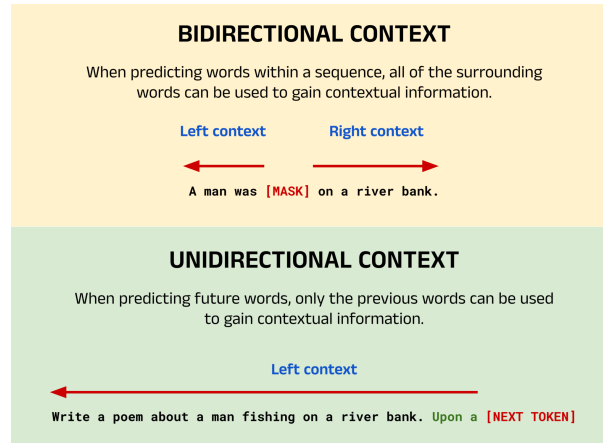


Figure 2.6. Bidirectional vs. unidirectional context. *Top:* in a bidirectional encoder (BERT), predicting a masked token [MASK] exploits both left and right context simultaneously. *Bottom:* in a unidirectional decoder (GPT), only the preceding tokens are available as context for predicting the next token. Image: [71].

The following paragraph describes the internal structure of BERT and the way its input is formed; both aspects bear directly on the embedding pipeline of Section 2.1, of which BERT-style encoders are the realisation.

Architecture and input representation. BERT-base consists of 12 stacked encoder layers, each with 12 attention heads and a hidden dimension of $d = 768$, for a total of 110M parameters; BERT-large scales this to 24 layers, 16 heads, and $d = 1024$ (340M parameters). As illustrated in Figure 2.7, the input to each layer is formed by summing three learned embedding components:

- **Token embedding:** each token $t_i \in \mathcal{V}$ is mapped to a dense vector via the embedding matrix $W_E \in \mathbb{R}^{|\mathcal{V}| \times d}$.
- **Positional embedding:** a learned vector encodes the absolute position of each token, restricting the maximum sequence length to 512 tokens (unlike the sinusoidal encodings of the original Transformer).
- **Segment embedding:** a vector E_A or E_B identifies which of the two input sentences each token belongs to, enabling pairwise sentence tasks.

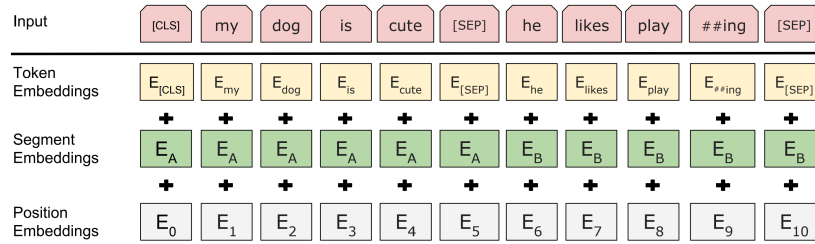


Figure 2.7. Input representation of BERT [2]. Each token is represented as the sum of three learned embeddings: the token embedding $E_{[\text{CLS}]}, E_{\text{my}}, \dots$ (yellow); the segment embedding E_A or E_B (green), distinguishing the two input sentences separated by [SEP]; and the position embedding $E_0, E_1, \dots$ (white), encoding the absolute position of each token in the sequence.

Pretraining objectives. BERT is pretrained on two self-supervised objectives. In *masked language modelling* (MLM), 15% of input tokens are replaced by a [MASK] symbol and the model is trained to recover them from context; unlike causal language modelling, MLM can exploit both left and right context simultaneously, directly leveraging the bidirectional attention of the encoder. In *next sentence prediction* (NSP), the [CLS] token is trained to predict whether two sentences are contiguous in the original corpus, reinforcing its role as a sequence-level summary used for classification and retrieval tasks.

Models used in this thesis. All six embedding models evaluated in this thesis are built on encoder-only architectures in the BERT tradition, differing primarily in scale, training data, and fine-tuning strategy. They are summarised in Table 2.1, which also defines the short aliases used to refer to each model throughout the text. A detailed account of their experimental configuration is deferred to Chapter 3. For the two proprietary models (Gemini, Azure), no peer-reviewed architecture paper is available; citations refer to the official technical reports and release documentation published by the respective providers.

Table 2.1. Embedding models evaluated in this thesis, with deployment provider and short alias used throughout the text. [†] Architecture not publicly disclosed.

Model (Alias)	Provider	Dim.	Characteristic
bge-m3 [45] (HF_bge)	Hugging Face	1024	XLNet encoder; dense/sparse/ColBERT heads; multilingual contrastive fine-tuning
multilingual-e5-large [46] (HF_e5)	Hugging Face	1024	XLNet encoder; instruction-tuned on multilingual text pairs (100+ languages)
nomic-embed-text [47] (Ollama)	Ollama (local)	768	BERT encoder with RoPE and MoE feed-forward layers; long-context capable
nemotron-embed-v1-1b [48] (OR_nvidia)	OpenRouter	4096	LLaMA decoder distilled into a bidirectional encoder; vision-language pre-training
gemini-embedding-001 [49] [†] (Gemini)	Google	3072	Proprietary; Matryoshka Representation Learning; variable output dimensionality
text-embedding-3-large [50, 51] [†] (Azure)	Azure OpenAI	3072	Proprietary; Matryoshka Representation Learning; high-dimensional isotropic space

These encoder-only architectures produce the embedding vectors studied throughout this thesis. The complementary family of generative models, which serves as the language backbone of the RAG pipeline, is introduced in the next section.

2.3 Generative Large Language Models

The two years following the original Transformer established the pretraining paradigm that still dominates the field. BERT [2] showed that an encoder pretrained on masked language modelling could be fine-tuned across a wide range of understanding tasks, while the GPT series [3] demonstrated the same for autoregressive generation. Between 2020 and 2021, models such as GPT-3 (175B parameters), T5 [43], and PaLM probed the relationship between scale and downstream performance, revealing emergent capabilities absent in smaller models. From 2022, instruction tuning and Reinforcement Learning from Human Feedback (RLHF) [34] became central, yielding models optimised for human interaction such as InstructGPT and ChatGPT. The 2023-2024 period was characterised by the proliferation of open-weight models (LLaMA, Mistral, Falcon) and a push towards longer context windows, multimodality, and chain-of-thought reasoning [35]. Since 2025 the dominant trend has been the *agentic turn*: models explicitly designed for multi-step reasoning, tool use, and autonomous task execution, as illustrated in Figure 2.8.


$$f_{\theta}(x_1, \dots, x_n) := \mathbb{P}_{\theta}(X_{n+1} = x_{n+1} \mid x_1, \dots, x_n).$$
$$\mathbb{P}_\theta(x_1, \dots, x_n) = \prod_{t=1}^n \mathbb{P}_\theta(x_t \mid x_1, \dots, x_{t-1}).$$

Text generation proceeds by sampling from this distribution one token at a time, each step conditioning on all previously generated tokens. The term *large* refers both to the number of parameters, typically ranging from hundreds of millions to hundreds of billions, and to the scale of the pretraining corpora. While historically some LLMs used encoder-decoder architectures (T5, BART), the dominant paradigm since GPT-3 has been decoder-only: models such as GPT-4, LLaMA, Gemini, and Claude all follow this design, which scales more efficiently and supports flexible in-context conditioning without a separate encoding phase [3].

Training

LLM training proceeds in two stages. The first is self-supervised *pretraining* on large unlabelled corpora; the second is task-specific *fine-tuning* on smaller annotated datasets.

Pretraining. The model is optimised using self-supervised objectives. For decoder-based models, the standard objective is *causal language modelling* (CLM): the model maximises the log-likelihood of each token given all preceding tokens,

$$\mathcal{L}_{\text{CLM}}(\theta) := \sum_{t=1}^n \log P_{\theta}(x_t \mid x_1, \dots, x_{t-1}).$$

For encoder-based models such as BERT, the objective is instead *masked language modelling* (MLM), as described in Section 2.2. Both objectives are entirely task-agnostic: the model acquires general-purpose linguistic representations without being optimised for any specific downstream application.

Fine-tuning. After pretraining, the model is adapted to specific tasks on smaller labelled datasets. A widely adopted approach is *instruction tuning* [35], where the model is trained on (*instruction, response*) pairs to follow natural language directives. Formally, fine-tuning maximises the likelihood of the target output conditioned on the instruction input:

$$\mathcal{L}_{\text{FT}}(\theta) := \sum_{(x,y) \in \mathcal{D}_{\text{tune}}} \log P_{\theta}(y \mid x),$$

where gradients are computed only over the output tokens y .

A further alignment stage, *Reinforcement Learning from Human Feedback* (RLHF) [34], trains a reward model on human preference judgements and uses it to steer the LLM towards responses that are helpful, harmless, and honest.

Despite the effectiveness of these training procedures, generative LLMs retain two structural limitations that are directly relevant to the RAG system studied in this thesis.

Limitations

Despite their capabilities, LLMs suffer from two structural limitations directly relevant to this thesis. First, their knowledge is *static*: the parameters encode information present in the training corpus up to a fixed cutoff date and cannot be updated without full retraining. Second, LLMs are prone to *hallucinations*: they generate fluent but factually incorrect content, because the training objective maximises token-level likelihood rather than factual accuracy [53].

Several strategies have been proposed to mitigate hallucinations. Prompt-engineering approaches instruct the model to cite sources, acknowledge uncertainty, or refuse to answer when evidence is insufficient. Fine-tuning on human-labelled preferences, via RLHF, can reduce overconfident claims. Knowledge grounding via external retrieval the RAG

paradigm described in the next section provides the model with verifiable context at inference time, constraining generation to information present in the retrieved documents and thereby reducing reliance on potentially outdated or fabricated parametric knowledge [4]. In the context of this thesis, RAG is the primary mitigation strategy, and its sensitivity to typographical corruption of the input query is the central empirical question.

2.4 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) is a framework that augments a generative LLM with a dynamic, external knowledge source accessed at inference time. Rather than relying solely on the information compressed into the model parameters during pretraining, a RAG system retrieves relevant documents from a corpus and supplies them to the LLM as additional context for generation [36, 4]. This directly addresses both structural limitations of Section 2.3: retrieved documents can reflect knowledge beyond the training cutoff, and they provide a verifiable factual grounding that reduces hallucinations.

Two terms appear throughout this section with distinct meanings and are clarified here to avoid ambiguity with the homonymous use of *query* in the attention mechanism of Section 2.2.

Remark 2.4.1 (Query vs. prompt). *A query is the user’s information need as originally formulated, a short natural-language string such as “how do I filter data in Rulex Factory?”. A prompt is the full input string presented to the LLM at generation time, which includes the system instruction, the retrieved context chunks, and the original query concatenated into a structured template. The query is thus a component of the prompt, but the two are not interchangeable: the embedding model operates on the query alone, while the LLM operates on the full prompt.*

The RAG pipeline

The RAG pipeline decomposes into four logical steps [36], illustrated in Figure 2.9:

- I. **Ingestion.** The documents in the knowledge base are split into smaller units of text called *chunks*, denoted c_i , which together form the retrieval corpus $\mathcal{C} := \{c_1, \dots, c_N\}$. Each chunk is passed through the embedding model E to obtain a dense vector $E(c_i) \in \mathbb{R}^d$ encoding its semantic content as a point in the latent space. The resulting vectors are stored in a *vector database* that supports approximate nearest-neighbour (ANN) search [54]. This phase is performed offline, once, and independently of any specific user query; as the knowledge base is updated, new chunks can be embedded and indexed incrementally without reindexing the entire corpus.
- II. **Retrieval.** The user query $q \in \Sigma^*$ is embedded by the same model E to obtain a query vector $E(q) \in \mathbb{R}^d$. The set of the k nearest neighbours of $E(q)$ in the corpus

$\mathcal{C}$, denoted $\mathcal{R}(q)$, is retrieved by minimising a distance δ on $\mathbb{R}^d$:

$$\mathcal{R}(q) := \arg \text{top-k}_{c_i \in \mathcal{C}} \delta(E(q), E(c_i)).$$

In standard RAG deployments, δ is cosine distance, the natural choice on the unit sphere $\mathbb{S}^{d-1}$ onto which contrastively trained models project their outputs (Section 1.1). This thesis evaluates retrieval stability under four distance functions (cosine, Euclidean, Manhattan, and Chebyshev), as detailed in Section 3.3. The retrieval step is entirely mediated by the geometry of the embedding space: two chunks are close if and only if they are semantically similar under the metric induced by E .

III. **Augmentation.** The retrieved chunks $\mathcal{R}(q)$ are concatenated with the original query into a structured prompt:

[Context: $\mathcal{R}(q)$] [Question: q] [Answer:]

The quality of the augmented context is upper-bounded by the quality of the retrieval step: no matter how capable the LLM, it cannot reason from documents it was never given.

IV. **Generation.** The augmented prompt is passed to the LLM f_θ , which generates the response conditioned on both the user query and the retrieved context:

$$\hat{y} \sim f_\theta(\cdot \mid q, \mathcal{R}(q)).$$

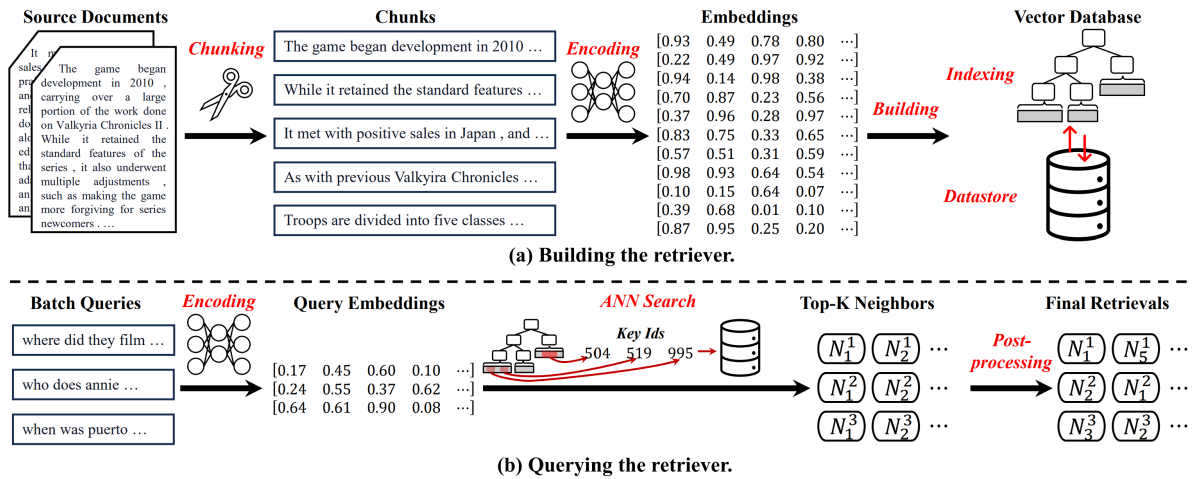


Figure 2.9. The four-step RAG pipeline [36]. Documents are chunked and indexed offline as embedding vectors in a vector database. At inference time, the query is embedded and the k nearest chunks $\mathcal{R}(q)$ are retrieved by nearest-neighbour search and supplied to the LLM as context for generation.

AI agents and agentic RAG

A standard RAG system executes a fixed, single-shot pipeline: one retrieval step, one generation step. This design breaks down when the task requires multi-step reasoning, the use of external tools, or iterative refinement of the retrieval strategy. *AI agents* address this by turning the LLM into an active decision-maker: rather than generating a single response, the agent operates in a loop, selecting actions from a space $\mathcal{A}$ that includes both text generation and *tool calls*, which are structured invocations of external functions such as web search, code execution, database queries, or API calls [38]. Tools are necessary because an LLM’s knowledge is static: without them, it cannot access real-time information or perform precise computation. Common tool categories are listed in Table 2.2.

Table 2.2. Common tools available to AI agents [38].

Tool	Description
Web search	Fetches up-to-date information from the internet, bypassing the training cutoff.
Retrieval	Queries an external vector database (RAG).
Code executor	Runs arbitrary code, enabling data processing.
Image generation	Produces images from text descriptions.
API interface	Interacts with external services (GitHub, databases, calendars, etc.).

The key enabling mechanism is the *ReAct* loop [37], which interleaves three step types: a **Thought** (internal reasoning trace), an **Act** (tool call), and an **Observation** (tool result fed back into context). This cycle allows the agent to update its plan mid-execution and correct errors that a purely generative model would hallucinate past, building on the *chain-of-thought* prompting paradigm [55].

In *agentic RAG* [36], retrieval is no longer a single fixed step but a dynamic decision within the agent loop: the agent may decompose a query into sub-queries, issue follow-up retrievals, or re-rank chunks via a reasoning step before generation. This is the dominant deployment pattern in production systems today; standard single-shot RAG serves here as a controlled experimental baseline against which more complex retrieval strategies can be compared.

Role of the embedding model in RAG

In both standard and agentic RAG, the retrieval step is entirely mediated by the embedding model E : the LLM receives the user query as part of the prompt, but for domain-specific knowledge bases its response depends critically on $\mathcal{R}(q)$, since the relevant information is not present in the model’s parametric knowledge.

A typographically corrupted query $\tilde{q}$ may shift $E(\tilde{q})$ away from the relevant region of the latent manifold, causing $\mathcal{R}(\tilde{q})$ to diverge from $\mathcal{R}(q)$: the top- k nearest neighbours

become semantically unrelated to the original intent, a failure the LLM cannot recover from since the relevant documents are simply absent from its context. In agentic RAG this risk compounds across iterations: a corrupted query vector at the first retrieval step can cascade through the entire ReAct loop [37], propagating an incorrect context to every subsequent reasoning and generation step. Figure 2.10 illustrates this mechanism on a concrete example drawn from the Rulex Platform documentation (version 1.4.x, 239 files [56]), using the `text-embedding-3-large` retriever (Azure/OpenAI).

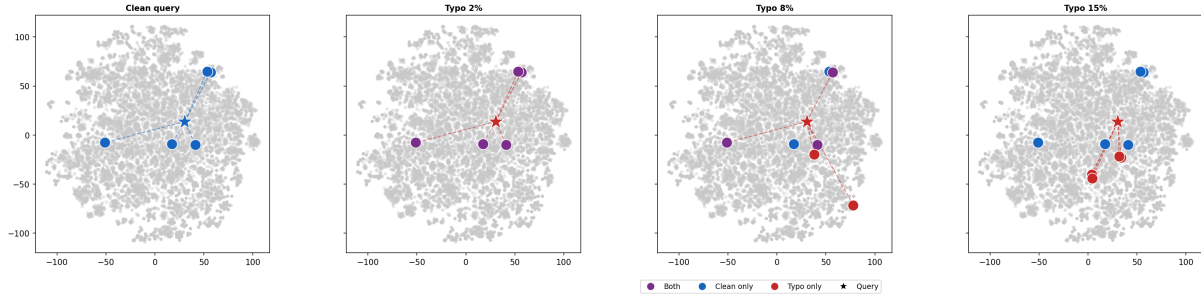


Figure 2.10. t-SNE projections of the Rulex Platform knowledge base (grey) with top- k retrieved chunks for the clean query (blue/purple) and its typo variant (red/purple); stars mark query vectors. Clean query: “How do I filter my data in Rulex Factory?” — Typo variants: $\rho=2\%$: “how do i filter my data in rulex factory?”; $\rho=8\%$: “how do i filter my data in rulex fatcory?”; $\rho=15\%$: “how do l dilter jy data in rukex fcatology?”. As ρ increases, $E(\tilde{q})$ drifts progressively and $\mathcal{R}(\tilde{q})$ diverges from $\mathcal{R}(q)$.

Remark 2.4.2. The apparent spatial distances in Figure 2.10 should be interpreted with care: *t*-SNE preserves local neighbourhoods but distorts global geometry, so two points that appear far apart in the 2D projection may not be so in the original high-dimensional space. In all cases, $\mathcal{R}(\tilde{q})$ contains the true top- k nearest neighbours of $E(\tilde{q})$; if they appear distant in the projection, it is because the query has drifted into a sparsely populated region of the latent manifold and the retriever is returning the least-far available results rather than genuinely close matches.

2.5 Summary

This chapter has introduced the three computational systems that together constitute the experimental pipeline of this thesis, tracing a path from the geometric representation of text to the end-to-end failure mode under typographical noise.

- **Section 2.1** developed the theory of embedding models. A model $E : \Sigma^* \rightarrow \mathbb{R}^d$ converts raw strings into vectors in a high-dimensional latent space, where proximity encodes semantic similarity. The tokenisation pipeline, the geometry of the latent manifold $\mathcal{M} \hookrightarrow \mathbb{R}^d$, and the contrastive training objectives that shape it were examined in turn. The section culminated in the mathematical reformulation of E as a Lipschitz map between metric spaces: the upper constant C of its bi-Lipschitz band controls stability, the lower constant c controls discriminability, and their ratio $\kappa = C/c$ is the distortion index that the experiments of Chapter 4 seek to estimate.
- **Section 2.2** described the Transformer architecture that underlies every model evaluated in this thesis. The self-attention mechanism, which computes contextualised token representations by weighting value vectors according to query-key compatibility, was examined in detail through the scaled dot-product formula and a worked example. The bidirectional encoder variant (BERT) provides the architectural backbone of the six embedding models of Table 2.1, while decoder-only models serve as the generative component of the RAG pipeline.
- **Section 2.3** introduced generative Large Language Models as the second computational component of the RAG pipeline. An LLM is a parametric function $f_\theta : \mathcal{V}^* \rightarrow \Delta(\mathcal{V})$ that models the conditional distribution over the next token; text generation proceeds by autoregressive sampling from this distribution. The two-stage training paradigm, pretraining on large unlabelled corpora followed by instruction tuning and RLHF, yields models that are fluent and instruction-following but structurally limited by a fixed knowledge cutoff and a tendency towards hallucination. Both limitations are directly addressed by the RAG framework: retrieved documents supply up-to-date factual grounding at inference time, constraining generation to verifiable content.
- **Section 2.4** connected the embedding model to the end-to-end RAG system. The four-stage pipeline (ingestion, retrieval, augmentation, generation) makes the role of E explicit: retrieval is entirely mediated by nearest-neighbour search in the embedding space, so any degradation of E under typographical noise propagates directly to the quality of the retrieved context and, through it, to the quality of the generated answer. In agentic RAG, this risk compounds across ReAct iterations, since a corrupted query vector at the first retrieval step can cascade through the entire reasoning loop. The t-SNE visualisation of Figure 2.10 illustrates this drift concretely on the Rulex Platform knowledge base, providing the empirical motivation for the three-level experimental hierarchy of Chapter 4: noise model, embedding displacement, and retrieval and generation degradation.

3 Problem Modeling and Experimental Methodology

This chapter formalises the three experimental modules that constitute the empirical core of the thesis, grounding each one in the mathematical framework developed in Chapters 1 and 2. While the two preceding chapters recalled established theory from the literature, the modeling choices, formalisations, and experimental design presented from this chapter onward constitute original contributions of this thesis, developed specifically to address the research questions stated below. It is structured as follows:

- **Section 3.1** states the three nested research questions addressed by the thesis: the effect of typographical noise on the character-level structure of a query, its propagation into the embedding space, and its downstream impact on retrieval quality in a RAG pipeline.
- **Section 3.2** formalises the typographical noise process: the Markov-chain noise channel introduced in Section 1.3 is instantiated as a character-level confusion matrix estimated from a real-world dataset, and the spectral diagnostics used to validate the fitted matrix are defined.
- **Section 3.3** models embedding stability under noise. The perturbation model is posed as a Lipschitz-continuity problem in the metric spaces described in Section 2.1. The distance degradation is formalised, and the dataset and evaluation metrics for the stability experiments are described.
- **Section 3.4** adapts the noise and stability models to the full RAG pipeline of Section 2.4: the retrieval backbone is treated as a noisy channel, the evaluation protocol follows the standard RAG benchmarking methodology.

3.1 Problem Statement

Let $E : \Sigma^* \rightarrow \mathbb{R}^d$ denote a text embedding function mapping a string over alphabet Σ to a d -dimensional real vector, and let d_{out} be a distance function on $\mathbb{R}^d$. Given a clean query $q \in \Sigma^*$ and a perturbed version $\tilde{q} \in \Sigma^*$ at Damerau-Levenshtein distance

$d_{\text{DL}}(q, \tilde{q}) = \delta > 0$, the central question of this work is whether E is *bi-Lipschitz* with respect to the metric pair $(d_{\text{DL}}, d_{\text{out}})$ as defined in Equation (2.1.2), i.e. whether there exist constants $0 < c \leq C < \infty$ such that

$$c \cdot \delta \leq d_{\text{out}}(E(q), E(\tilde{q})) \leq C \cdot \delta \quad (3.1.1)$$

for all perturbation pairs, and what the empirical values of c and C are across different embedding models, perturbation types, and output-space distance functions. The two constants have distinct and complementary readings, already introduced in Definition 1.1.4 and worth restating here in the terms of this experiment. The upper constant C answers the question “*how far can a typo push my query in the latent space?*”: it bounds the damage. The lower constant c answers the opposite question, “*are two genuinely different queries still kept apart?*”: it bounds the collapse. A model can fail in either direction. If C is large, a harmless typo is amplified into a large displacement and retrieval breaks; if c is near zero, distinct queries are mapped almost on top of one another and retrieval becomes arbitrary. As in Chapters 1 and 2, the ratio $\kappa := C/c$ summarises both failures in a single dimensionless number. As in Chapters 1 and 2, the ratio $\kappa := C/c$ summarises the distortion of the embedding. A model with C close to c behaves nearly as an isometry; a model with $C \gg c$ amplifies minor corruptions into large, query-dependent displacement, degrading retrieval even when the underlying information need is unchanged.

Definition 3.1.1. *To estimate these constants empirically, I introduce the Normalised Embedding Displacement (NED): the pointwise ratio of output-space displacement to input-space edit distance,*

$$\hat{L}_i := \frac{d_{\text{out}}(E(q), E(\tilde{q}^{(i)}))}{d_{\text{DL}}(q, \tilde{q}^{(i)})}, \quad (3.1.2)$$

computed for each perturbed sample $\tilde{q}^{(i)}$ with $d_{\text{DL}} > 0$. Being dimensionless, the NED is comparable across queries of different lengths and output distances of different scales. Aggregating $\hat{L}_i$ within and across queries yields a family of four robustness indices (NED, U-NED, L-NED, and the Distortion Index) whose precise definitions are given in Section 3.3.

The thesis addresses three nested research questions:

1. **Noise structure.** What are the statistical properties of the character-level confusion channel estimated from real-world spelling corpora, and which characters are disproportionately error-prone?
2. **Embedding stability.** How do the empirical Lipschitz and bi-Lipschitz constants of each embedding model vary with noise intensity and output-space metric? At what intensity does the displacement curve saturate?
3. **Retrieval impact.** Does embedding displacement under typographical noise degrade retrieval and generation quality in a RAG pipeline, and which embedding models are most resilient?

The evaluation metrics that operationalise questions 2 and 3 are defined in Sections 3.3 and 3.4 respectively. The full estimation pipeline is illustrated in Figure 3.1.

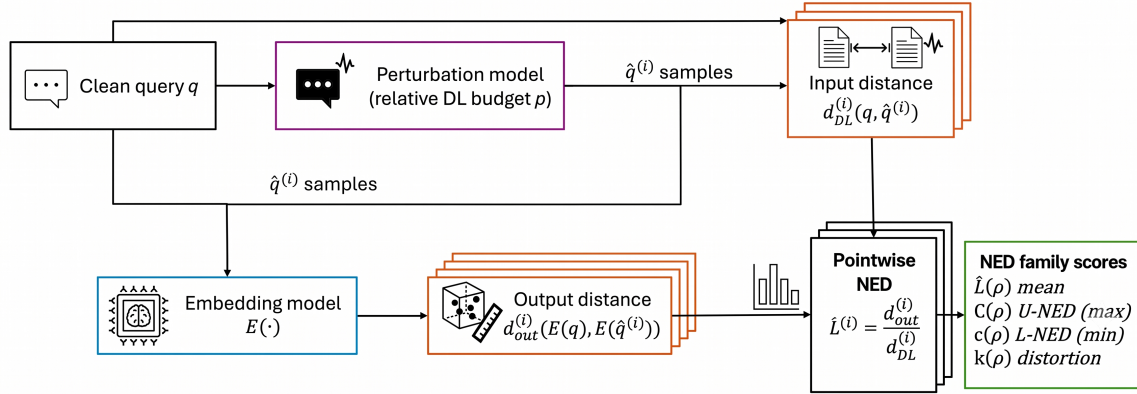


Figure 3.1. Overview of the NED estimation pipeline. A clean query q is perturbed N_{typo} times at relative DL budget p ; each perturbed variant is embedded and the ratio of output-space distance to Damerau-Levenshtein distance yields a pointwise NED estimate $\hat{L}_i$ (Equation 3.1.2). After filtering zero-DL samples, within-query and cross-query aggregation produce the four robustness indices (NED, U-NED, L-NED, and Distortion Index) defined in Section 3.3.

3.2 Typographical Noise

The first experimental module characterises the noise process itself: given a corpus of real misspellings, I estimate a stochastic character-confusion model, verify its theoretical properties, and use it as the common noise source for all downstream experiments. This section describes the perturbation model and its dataset, and defines the spectral diagnostics used to validate the fitted matrix.

Modeling

Realistic query corruption cannot be captured by a uniform random mutation channel: actual human errors concentrate on specific character pairs shaped by finger proximity, keystroke timing, and phonetic similarity. The noise model adopted here combines three complementary mechanisms, each targeting a distinct error source, into a single character-level channel. Together they cover all four canonical Damerau-Levenshtein error types: substitution, deletion, insertion, and transposition. All three mechanisms operate exclusively on alphabetic characters and spaces ($\Sigma_0 := \{\mathbf{a}, \dots, \mathbf{z}, _ \}$); digits, punctuation, and any other symbols are left unchanged.

- **QWERTY substitution.** This operation encodes the spatial structure of the keyboard. Character $c_i \in \Sigma_0$ is replaced by a key drawn from its QWERTY adjacency neighbourhood. With probability 0.70 the result is a physically adjacent

key (substitution); with probability 0.15 the character is deleted (deletion); with probability 0.15 it is doubled (insertion).

- **Markov substitution.** A first-order Markov chain with state space $\mathcal{S} := \Sigma_0 \cup \{\varepsilon\}$, where ε is a fictitious gap symbol, is estimated from the spelling-error corpora presented in the dataset subsection below. Its transition matrix $P \in [0, 1]^{|\mathcal{S}| \times |\mathcal{S}|}$ is row-stochastic and smoothed via Laplace regularisation, and encodes corpus-level confusion statistics: entry Q_{ij} is the empirical probability that character $i \in \Sigma_0$ is realised as character j . At generation time, each character $c_i \in \Sigma_0$ is independently replaced by a sample drawn from row c_i of P : sampling $j \neq c_i$ in Σ_0 produces a substitution, sampling $j = \varepsilon$ produces a deletion of c_i , and sampling any $j \in \Sigma_0$ from row ε produces an insertion.
- **Transposition.** Characters c_i and c_{i+1} are swapped, provided both belong to Σ_0 . This models the most frequent keystroke timing error, in which two adjacent fingers complete their strokes in the wrong order, and is the only error type not expressible as a single substitution, deletion, or insertion.

The three mechanisms are complementary by design: QWERTY substitutions capture physical proximity on the keyboard, Markov substitutions capture the statistical regularities of the error corpus, and transpositions capture timing errors between adjacent keystrokes. Their equal weighting reflects the absence of ground-truth data on relative error frequencies in short query texts; in the absence of such data, equal weighting is the least-biased choice.

Dataset and Evaluation Metrics

The noise model is fitted on two complementary spelling error corpora. The two sources share the same format but differ in origin, writing medium, and the type of speller they represent; combining them broadens the coverage of the confusion matrix and reduces the risk of corpus-specific bias. An example of their format is given in Figure 3.2.

- **The Birkbeck corpus.** The Birkbeck spelling error corpus [57] contains 36,133 misspellings of 6,136 words, compiled by Mitton at Birkbeck College and deposited in the Oxford Text Archive. The errors were collected from native-speaker spelling tests and free-writing samples by schoolchildren, university students, and adult literacy learners. Each correct word is preceded by a dollar sign and followed by its misspellings, one per line, without duplicates. The corpus spans a wide range of speller abilities, from mild transpositions to severe corruptions in which only a single letter of the target survives, ensuring that the resulting confusion matrix covers both common and rare substitution patterns.
- **The Wikipedia misspellings corpus.** The Wikipedia misspellings list [58] contains 2,455 misspellings of 1,922 words, drawn from errors actually committed by Wikipedia editors and maintained as a community resource for automated

spelling correction across the encyclopaedia. Unlike the Birkbeck corpus, which records handwritten errors by students of varying ability, the Wikipedia list reflects the typing habits of adult, digitally literate writers working under self-imposed editorial standards. Its errors are therefore more systematic and phonetically motivated, with a higher proportion of near-misses (single-character substitutions and omissions) relative to catastrophic corruptions. A single misspelling may appear as the realisation of several correct words: for example, *buring* is listed as a misspelling of *burin*, *burning*, *burying*, and *during*, reflecting the one-to-many structure of the human confusion channel.

Corpus format and sample entries

Birkbeck handwritten errors

\$through	\$achieve	\$some	\$rough	\$necessary	\$sometimes
tho	acheive	same	raf	necessary	some_time
threw	acheve	sane	ruff	neccesary	sume_time
thought	achi	so	rouch	neccesary	somtimes
throu			roughe		somtimes
throw			roth		some_time's
rowit			roagh		sumetimes

Wikipedia keyboard-typed errors

\$through	\$achieve	\$some	\$European	\$address	\$rhythm
through	acheive	smoe	European	addres	rhythem
thru	achive	soem	Eurpean	adres	rhythim
			Eurpoean	adress	rythem
					rythim
					rythm

Figure 3.2. Sample entries from the two spelling corpora used in this work. Each correct target (prefixed by \$) is followed by its attested misspellings. The three leftmost columns show words present in *both* corpora, enabling direct comparison across writing media; the remaining columns show words unique to each source. Birkbeck [57] entries tend to exhibit deep phonetic substitutions and morphological fragmentation (e.g. *rough*→*raf*, *sometimes*→*sum_time*), consistent with uncertainty at the point of handwriting. Wikipedia [58] entries are generally closer to the target in edit distance, though the underlying causes (whether attentional, motor, or orthographic) remain heterogeneous.

The character statistics needed to fit the Markov noise model are extracted from these corpora through pairwise sequence alignment. The following paragraph describes this extraction procedure and the two corpus-combination strategies available.

Character Transition Matrix. To extract character-level statistics, each (c, m) correct-misspelled word pair from both corpora is aligned via the Needleman-Wunsch

global sequence alignment algorithm [24], originally developed for protein sequence comparison. Prior to alignment, all strings are lowercased, so the analysis is case-insensitive and the state space $\mathcal{S}$ contains only the 26 lowercase letters plus the space character and the gap symbol ε . The algorithm then proceeds to fill an $(n+1) \times (m+1)$ dynamic programming table $\mathbf{D}$ under the recursion

$$D_{i,j} := \max \begin{cases} D_{i-1,j-1} + s(c_i, m_j) & \text{(match/substitution)} \\ D_{i-1,j} - 1 & \text{(deletion)} \\ D_{i,j-1} - 1 & \text{(insertion)} \end{cases}$$

with $s(c, m) := +1$ if $c = m$ and -1 otherwise, and boundary conditions $D_{i,0} := -i$, $D_{0,j} := -j$. The optimal alignment is recovered by traceback from cell (n, m) , and each aligned pair (c, m) contributes a count n_{cm} to the raw confusion matrix.

The two corpora can be combined via two distinct strategies:

- The **pooled strategy** (`normalize=False`). The alignment pairs from all sources are concatenated before the count matrix is computed, so that a corpus with more entries has proportionally greater influence on the final matrix. This is the empirically driven choice: it weights error types by their observed frequency across the full available sample.
- The **averaged strategy** (`normalize=True`). A separate transition matrix is estimated from each corpus and the per-entry arithmetic mean is taken. This balances the contribution of the two sources regardless of their size, preventing the larger Birkbeck corpus from dominating the result and ensuring that the more systematic Wikipedia errors retain their influence.

The pooled matrix is used as the primary noise source throughout the experiments. The choice to weight Birkbeck more heavily is motivated by the nature of its errors: being collected from handwritten spelling corrections, Birkbeck captures *cognitive* confusions, that is, errors arising from imperfect phonological or orthographic knowledge rather than from finger mechanics. The Wikipedia corpus, by contrast, includes a non-negligible fraction of mechanical keyboard errors, which are already modelled by the dedicated QWERTY substitution channel (Section 3.2). Retaining them in the Markov channel would double-count the mechanical component at the expense of the cognitive one. The pooled matrix, dominated by Birkbeck, therefore provides a confusion distribution that is representative of the cognitive error layer, while the keyboard layer is handled orthogonally and independently. The averaged matrix is retained as an ablation baseline to assess the sensitivity of the results to the corpus-combination strategy.

In both cases the raw count matrix is row-normalised with Laplace smoothing:

$$Q_{cm} := \frac{n_{cm} + \lambda}{\sum_{m'} (n_{cm'} + \lambda)}, \quad c, m \in \mathcal{S}, \quad \lambda := 10^{-6}. \quad (3.2.1)$$

Smoothing eliminates zero-probability transitions and guarantees that Q is strictly positive. This has a concrete consequence, not merely a formal one: strict positivity

means every character can be confused with every other in a single step (irreducibility, Definition 1.3.5) and every character has a non-zero chance of being transmitted correctly (aperiodicity, via the self-loops $q_{cc} > 0$, Proposition 1.3.2). These are exactly the two hypotheses of the ergodic theorem (Theorem 1.3.5), and they are what allow the noise channel to be summarised by a single stationary vector $\boldsymbol{\pi}$ rather than by a family of starting-state-dependent behaviours. Without smoothing, a character never observed as a typo source would become an absorbing state (Definition 1.3.5): once the chain reached it, corruption would stop there permanently, and the channel would no longer be a well-behaved noise source.

Three spectral diagnostics are applied to the fitted transition matrix to validate its theoretical properties before it is used as a noise source. These diagnostics directly apply the Markov chain theory of Section 1.3: the ergodicity analysis uses the communicating class decomposition (Definition 1.3.5), the stationary distribution is the dominant left eigenvector guaranteed by Corollary 1.3.4, and the confusability index is a per-character summary derived from the diagonal of Q .

- The **ergodicity analysis** computes the strongly connected component (SCC) decomposition of the directed graph of Q . For the smoothed matrix a single SCC is expected by construction; for the raw (unsmoothed) matrix the decomposition reveals transient states and potential closed classes. The gap symbol ε is the most likely candidate for a closed class in the raw chain: if no corpus entry records a recovery from a deletion, then $Q_{\varepsilon,c} = 0$ for all $c \neq \varepsilon$, making deletion an absorbing event under the raw model.
- The **stationary distribution** $\boldsymbol{\pi}$, defined by $\boldsymbol{\pi}^\top Q = \boldsymbol{\pi}^\top$ with $\boldsymbol{\pi}^\top \mathbf{1} = 1$, is computed as the dominant left eigenvector of Q and compared with the empirical letter-frequency distribution of English [59]. A systematic excess of π_c over the corpus frequency of character c indicates that the typo channel over-represents c as a mutation target, potentially biasing the perturbation experiments toward specific regions of the latent space.

Convergence to $\boldsymbol{\pi}$ is validated via a Monte Carlo simulation: a single chain of length $M := 2 \times 10^4$ is run, and at each step m the occupation frequency $N_m(a)/m$ (the fraction of steps spent in state a) is recorded. By Remark 1.3.8, $N_m(a)/m \rightarrow \pi_a$ almost surely as $m \rightarrow \infty$. The empirical *mixing time* is defined operationally as

$$t_{\text{mix}} := \min \left\{ m : \max_a \left| \frac{N_m(a)}{m} - \pi_a \right| < 10^{-2} \right\},$$

i.e. the first step at which every character's empirical frequency lies within 1% of its stationary value, regardless of the starting state. The 1% threshold is a conventional engineering tolerance, analogous to the total variation tolerance used in formal mixing-time theory [18]. A finite t_{mix} confirms empirically what Theorem 1.3.5 guarantees in theory: the chain converges to $\boldsymbol{\pi}$ regardless of where it starts, and π_c is therefore not an arbitrary vector but the long-run frequency with which character c appears if the noise channel is applied repeatedly. In the language of Chapter 1, $\boldsymbol{\pi}$ is the unique fixed point of the map $\mu \mapsto \mu Q$

(Definition 1.3.10): the distribution that no longer changes when the channel is applied again. Verifying that the simulated chain actually reaches it is what licenses the comparison against English letter frequencies below: without convergence, π_c would be a formal eigenvector with no operational meaning, and any discrepancy from natural-text frequency could not be attributed to a bias of the noise model.

It is important to note that t_{mix} doesn't constrain the independence of the $N_{\text{typo}} = 500$ perturbation samples used in the embedding experiments. Those samples are drawn independently: each is generated fresh from the clean query by applying the noise channel once, not by advancing a sequential chain. Their independence therefore holds by construction and does not require t_{mix} to be smaller than any experimental budget.

- The **confusability indices**, defined below, summarise the per-character error rate of the fitted channel. Two normalised variants are introduced to separate the distributional bias of the typo corpus from intrinsic keyboard or phonetic difficulty.

Definition 3.2.1 (Confusability indices). *Let Q be the fitted confusion matrix and π its stationary distribution. For each character $c \in \Sigma_0$, the raw confusability index is*

$$\chi(c) := 1 - Q_{cc},$$

measuring the probability that c is not transmitted correctly under the noise channel. Two frequency-normalised variants are defined. The stationary-normalised confusability

$$\tilde{\chi}(c) := \frac{\chi(c)}{\pi_c}$$

identifies characters confused more often than their stationary frequency alone would predict, highlighting keyboard-proximity effects and phonetic traps. The corpus-normalised confusability

$$\hat{\chi}(c) := \frac{\chi(c)}{f(c)},$$

uses the empirical frequencies of written English $f(c)$ [59] in the denominator rather than the corpus-derived π_c : a character with $\hat{\chi}(c) \gg 1$ is disproportionately error-prone given how common it is in everyday writing.

Remark 3.2.1. *While $\tilde{\chi}(c)$ measures excess confusion relative to how often c appears as a typo target in the noise model, $\hat{\chi}(c)$ measures it relative to how often c occurs in natural text. Comparing the two therefore separates two distinct sources of confusability: distributional bias in the typo corpus versus intrinsic keyboard or phonetic difficulty. High-confusability characters under both normalisations drive retrieval degradation at low noise levels and are discussed further in Chapter 4.*

3.3 Embedding Stability

The embedding stability module formalises the relationship between input-space perturbation and output-space displacement through the Normalised Embedding Displacement (NED) index introduced in Section 3.1. The goal is to characterise how each embedding model responds to controlled orthographic noise and to identify which configurations maintain low NED across the full intensity range. This section describes the experimental setup and defines the robustness indices and displacement thresholds used to summarise the shape of the NED curve. The experimental setup is described first, covering the perturbation budget, the noise channel, the embedding providers and the output-space distance functions used to measure displacement; the evaluation metrics and the saturation analysis follow in the second subsection.

Modeling

The following paragraphs describe the four components of the experimental setup: the perturbation budget axis, the character-level noise channel, the set of embedding providers, and the output-space distance functions used to measure displacement.

Intensity axis. All experiments are parameterised by the *relative Damerau-Levenshtein budget* $\rho \in (0, 1]$ (Definition 1.3.14), which targets $\lfloor \rho \cdot |q| \rfloor$ edits on the input query q . A perturbed query at budget ρ is produced by an accept/reject sampler: up to 200 candidates are drawn from the character-level noise channel and the one whose DL distance to the original is closest to the target is retained. The following values are tested:

$$\rho \in \{0.01, 0.02, \dots, 0.10, 0.15, 0.20, 0.25\}.$$

This parameterisation ensures cross-query comparability: $\rho = 0.10$ always corrupts approximately 10% of the characters regardless of query length, avoiding the length-dependent variance of a per-character mutation rate (Remark 1.3.10).

Perturbation channel. Given a clean query $q = c_1 c_2 \dots c_n$, the character-level channel applies the attenuated noise channel M (Definition 1.3.13) iteratively until $\text{DL}(q, \tilde{q}) = \lfloor \rho \cdot |q| \rfloor$. Each mutation is drawn uniformly from three operations: substitution, insertion, and deletion. This is the most controlled setting for isolating the effect of orthographic noise on embedding displacement while preserving the lexical identity of most tokens.

The perturbed queries produced by this channel are then embedded by each of the six models in the evaluation.

Embedding providers. Six embedding models are evaluated, spanning output dimensionalities from 768 to 4096 and covering local open-weight deployments (Ollama, HuggingFace) and proprietary API providers (Azure, Google, OpenRouter); the full list with architectural details is given in Table 2.1 (Section 2.2). Local open-weight models

run on-premise at no marginal cost; API-based models are queried with rate-limit-aware batching and exponential back-off on HTTP 429 responses. A persistent MD5-keyed JSON cache avoids redundant API calls across runs.

For each (query, ρ) pair, $N_{\text{typo}} := 500$ independent perturbed queries are generated and embedded. Two filtering steps are applied before computing NED (3.1.2) to guarantee the validity of the ratio $\widehat{L}_i$. First, samples with $d_{\text{DL}}(q, \tilde{q}) = 0$ are excluded: for short queries or very small ρ , the target edit count $\lfloor \rho \cdot |q| \rfloor$ can be zero and the sampler may return the original query unchanged; these carry no perturbation signal and would produce an undefined ratio. Second, samples for which the provider returns a malformed embedding vector are dropped. A *malformed* vector here means one that cannot enter the NED computation at all: a response of the wrong dimensionality, a vector containing NaN or Inf entries, or an empty response. Any of these would propagate an undefined value into d_{out} and hence into $\widehat{L}_i$, contaminating the aggregation.

It is worth being explicit about what this filter is *not*. It does not remove queries the provider found objectionable, and it is not a content filter of any kind: unlike generative LLMs, embedding models apply no moderation layer, since their only output is a fixed-length numerical vector and there is no text to moderate. The filter targets three purely infrastructural failure modes. First, a transient network or encoding error not absorbed by the exponential back-off retry logic. Second, an empty input string: at high deletion rates the perturbation channel can, in principle, delete every character of a short query, and an empty string has no embedding. Third, a query whose token count exceeds the model’s context limit after repeated insertions, causing the provider to reject it.

The filter is therefore a safeguard against pipeline artefacts, not a source of selection bias, and its inactivity is itself an experimental result: as reported in Section 4.2, neither this filter nor the zero-DL filter removed a single sample in practice, so $|\mathcal{V}_q| = N_{\text{typo}} = 500$ holds uniformly and all reported statistics are computed on the full sample.

Once embedding vectors are obtained, displacement is quantified under multiple output-space distances to capture different geometric aspects of the perturbation.

Output-space distance functions. Displacement is measured under the four distances described in Example 1.1.2 and Definition 1.1.16, providing a multi-geometric view of the embedding space. All four are topologically equivalent on $\mathbb{R}^d$ (Theorem 1.1.4) and produce qualitatively concordant displacement profiles; they are retained because they differ in scale and in how individual coordinate deviations are weighted, providing complementary diagnostic information about the geometry of the displacement.

All distance metrics are computed directly on the embedding vectors as returned by each provider, without additional ℓ_2 -normalisation. Under this convention, cosine and Euclidean distances are not equivalent even for providers that approximately normalise their output: cosine distance is sensitive only to the angular component of the displacement $E(q) - E(\tilde{q})$, while Euclidean distance conflates angular and magnitude components. The two metrics therefore constitute genuinely distinct measurements, and their differential behaviour under typographical perturbations is part of the empirical

contribution of this thesis.

Dataset and Evaluation Metrics

With the experimental setup fixed, the remaining task is to define the query set, the robustness indices, and the saturation analysis that summarise the NED curves produced by each (model, metric) pair.

Twenty queries spanning five semantic and structural categories are used throughout all perturbation experiments (Table 3.1):

- **Typo-sensitive** queries contain words at DL distance 1 from a different real word, so that a single mutation can silently flip the meaning without producing an out-of-vocabulary token.
- **Lexically ambiguous** queries contain words with multiple unrelated senses.
- **Technically specific** queries are domain-specific informational requests whose correct answer exists in the knowledge base.
- **Benchmark** queries are short factual questions drawn from standard information retrieval evaluation sets [60, 61].
- **Structurally borderline** queries are syntactically unusual but semantically coherent, probing robustness at the boundary of the model’s training distribution.

Table 3.1. Query set used in all perturbation experiments, grouped by category.

#	Category	Query
1	Typo-sensitive	the cat sat on the mat
2	Typo-sensitive	I need to check the current
3	Typo-sensitive	the quick brown fox jumps over the lazy dog
4	Typo-sensitive	the gale blew the bale across the pale field
5	Lexically ambiguous	what is the best way to handle a bank overflow
6	Lexically ambiguous	how to train ai smart python
7	Lexically ambiguous	the current model runs on the latest patch
8	Lexically ambiguous	she took the right turn and left the bar
9	Technically specific	what can you do with a degree in mathematics besides suffering?
10	Technically specific	why should a pesto recipe with walnuts be considered a crime?
11	Technically specific	why does bread go stale faster in the fridge than on the counter
12	Technically specific	should i apply for the open position before the deadline
13	Benchmark [61]	what is the capital of France
14	Benchmark [60]	how does gradient descent converge?
15	Benchmark [61]	who is the first person who went to the moon
16	Benchmark [60]	what is a fandango
17	Structurally borderline	44 cats in a row of six with remainder of two
18	Structurally borderline	why are a donut and a mug the same?
19	Structurally borderline	why is the word short longer than the word long
20	Structurally borderline	how i met your mother who is the mother

The NED family of robustness indices formalises the relationship between input-space perturbation and output-space displacement, operationalising the bi-Lipschitz framework of Section 3.1 as concrete computable quantities. Four indices are computed from the same pointwise ratios $\hat{L}_i$, and it is worth stating in advance what each of them is for, since they answer four different questions: a reader interested in only one of those questions may read only the corresponding index. Two complementary analyses are then carried out: a stability analysis based on the empirical bi-Lipschitz band, and a saturation analysis that locates the noise level at which the displacement curve flattens.

The stability analysis proceeds in two aggregation stages.

Stage 1. For each query q at budget ρ , within-query statistics are computed over the valid sample set $\mathcal{V}_q$ introduced above:

$$\bar{L}_q(\rho) := \frac{1}{|\mathcal{V}_q|} \sum_{i \in \mathcal{V}_q} \hat{L}_i, \quad L_q^+(\rho) := \max_{i \in \mathcal{V}_q} \hat{L}_i, \quad L_q^-(\rho) := \min_{i \in \mathcal{V}_q} \hat{L}_i. \quad (3.3.1)$$

Stage 2. Aggregating across the query set $\mathcal{Q}$ yields four indices:

- **NED** $\hat{L}(\rho)$: mean amplification,

$$\hat{L}(\rho) := \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \bar{L}_q(\rho). \quad (3.3.2)$$

Each query contributes with equal weight regardless of its length or valid sample count, ensuring that short queries do not underrepresent the mean.

It answers the question: on average, how much does a typo move my query? This is the index to consult when comparing the typical, expected cost of noise across models. Its weakness is the weakness of every mean: it says nothing about the worst case, and a model can have an excellent mean while failing catastrophically on a minority of queries.

- **U-NED** $C(\rho)$: empirical upper Lipschitz constant,

$$C(\rho) := \max_{q \in \mathcal{Q}} L_q^+(\rho). \quad (3.3.3)$$

No observed pair $(q, \tilde{q}^{(i)})$ produces a ratio exceeding $C(\rho)$ at budget ρ .

It answers the question: *what is the worst that can happen?* As the empirical counterpart of the upper Lipschitz constant of Definition 1.1.4, it is the index to consult from a worst-case perspective. It is also directly comparable to the isometry threshold 1: a value $C > 1$ means that at least one perturbation produced a displacement in the latent space *larger* than the number of characters edited, that is, the model amplified the noise rather than absorbing it.

- **L-NED** $c(\rho)$: empirical lower Lipschitz constant,

$$c(\rho) := \min_{q \in \mathcal{Q}} L_q^-(\rho). \quad (3.3.4)$$

The pair $(c(\rho), C(\rho))$ constitutes the *empirical bi-Lipschitz band* of E , providing an empirical certificate for inequality (3.1.1).

It answers the question: *can the model ever fail to distinguish a perturbed query from its clean version?* A near-zero c means some perturbed query was mapped essentially on top of the original. This is not automatically undesirable, since a robust model *should* ignore harmless noise, but it is the quantity that certifies the lower bound of (3.1.1), and a c collapsing to zero is the empirical signature of the *metric collapse* described in Section 2.1.

- **Distortion Index** $\kappa(\rho)$:

$$\kappa(\rho) := \frac{C(\rho)}{c(\rho)}, \quad (3.3.5)$$

defined for $c(\rho) > 0$. Embeddings are ranked by their mean per-query distortion $\bar{\kappa}(\rho^*) := \frac{1}{|\mathcal{Q}|} \sum_q L_q^+(\rho^*)/L_q^-(\rho^*)$ at a fixed reference budget $\rho^* := 0.08$.

It answers the question: *does the model treat all queries in the same way?* A value close to 1 indicates uniform amplification across all inputs; large values reveal query-dependent sensitivity that mean-only analyses cannot detect. Two models with identical mean NED can differ by an order of magnitude in κ : one displaces every query by roughly the same amount, the other is nearly isometric on most queries and highly unstable on a few. Being a ratio of two quantities expressed in the same units, κ is dimensionless, and can therefore be compared directly across providers of different embedding dimension and across distance functions of different scale, where $\kappa = 1$ denotes a perfect isometry. It is the only index in the family that is comparable on a common scale, which is why it is adopted for the robustness ranking of Chapter 4.

The four indices are complementary rather than redundant, and the interest of the experimental results lies precisely in the cases where they disagree: a provider with a low mean NED but a high κ is *on average* stable and *occasionally* fragile, a distinction with direct practical consequences that no single number can express.

The four indices above characterise the shape of the NED curve globally. The saturation analysis that follows instead locates the specific budget level at which the curve transitions from rapid growth to a plateau. This matters for a practical reason. If displacement grew linearly with the edit count, the amount of noise a system could tolerate would be a matter of degree, and any threshold would be arbitrary. If instead the curve saturates, i.e. if most of the total displacement is already incurred after very few edits, then there exists a *critical region* of small ρ where robustness is decided, and everything beyond it is irrelevant because the damage is already done. The displacement thresholds are designed to answer exactly this: *how many edits does it take before the model has already suffered most of the harm it is ever going to suffer?* A ρ_{50} of 0.02 means half of the total displacement is absorbed by a single edit on a 50-character query; a ρ_{50} of 0.15 means the model degrades gracefully. The reader interested in choosing a tolerable noise level for a deployed system should read this quantity and not the NED family.

Saturation analysis. For each query q at budget ρ , let

$$\bar{d}_q(\rho) := \frac{1}{|\mathcal{V}_q|} \sum_{i \in \mathcal{V}_q} d_{\text{out}}(E(q), E(\tilde{q}^{(i)})) \quad (3.3.6)$$

denote the mean embedding distance over valid samples. The curve $\rho \mapsto \bar{d}_q(\rho)$ is typically concave-increasing: it grows rapidly at low intensity and flattens as additional corruption ceases to displace the query vector further in the latent space, a behaviour referred to throughout this thesis as *metric saturation*.

To summarise the saturation profile quantitatively, three *displacement thresholds* are defined as follows.

Definition 3.3.1 (Displacement threshold). *For $\alpha \in \{0.50, 0.75, 0.90\}$, the displacement threshold $\rho_\alpha(q)$ for query q is the first budget at which the embedding has accumulated at*

least fraction α of its total observed displacement:

$$\rho_\alpha(q) := \min \left\{ \rho_j : \bar{d}_q(\rho_j) \geq \alpha \cdot \max_k \tilde{d}_q(\rho_k) \right\}, \quad (3.3.7)$$

where $\tilde{d}_q(\rho_k) := \max_{j \leq k} \bar{d}_q(\rho_j)$ is the cumulative maximum of $\bar{d}_q$, applied to handle small non-monotonicities in the empirical curve.

In particular, ρ_{90} serves as a practical proxy for the saturation point: beyond it, additional corruption contributes at most 10% of the total observed displacement range.

The threshold is aggregated across the query set by the median:

$$\rho_\alpha := \text{median}_{q \in \mathcal{Q}} \rho_\alpha(q).$$

The median is preferred over the mean because a single outlier query would otherwise distort the aggregate; the interquartile range of $\rho_\alpha(q)$ across queries, reported as whiskers in the summary barplots, quantifies query-to-query variability.

The quantitative saturation profile is complemented by a geometric analysis in Chapter 4: t-SNE visualisations of the perturbed embedding clouds show the centroid trajectory as ρ increases, making it possible to identify visually the budget at which the centroid enters the dense collapse region and ceases to move significantly. The two analyses are designed to be read together: the thresholds provide a scalar summary of when saturation occurs, while the t-SNE figures provide evidence of *where* in the latent space the model saturates and confirm that the collapse is query-agnostic at high corruption budgets.

3.4 RAG Evaluation

The third experimental module inserts the perturbed queries into a complete RAG pipeline and measures whether embedding displacement translates into degradation of retrieved content and generated answers. This section describes the pipeline architecture and indexing strategy, the documentation corpus, and the two metrics used to quantify retrieval and generation robustness. The first subsection describes the pipeline architecture; the second defines the knowledge base and the evaluation metrics.

Modeling

The same six embedding providers of Table 2.1 are used as retrieval backbones. The generation step is fixed to a single LLM, Azure `gpt-4o-mini`, across all configurations, to isolate the effect of the embedding model on retrieval quality and avoid multiplying API costs by the number of generation models.

The RAG pipeline is an agent-based system built on LangChain [62] and ChromaDB [63], designed to be provider-agnostic: embedding models and LLMs are interchangeable through a unified factory interface without altering any other component of the stack. Three operational phases are distinguished.

1. **Indexing and ChromaDB construction.** Documents are loaded through a format-aware pipeline that handles plain-text and PDF files with standard recursive character splitting, and HTML files with a dedicated link-aware strategy introduced by [52]. HTML documents are split at paragraph-anchor boundaries into logical sections before character-level chunking is applied within each section; outgoing hyperlinks are extracted and stored in chunk metadata to enable the graph traversal described below. Chunking hyperparameters are listed in Table 3.2.

Table 3.2. Chunking hyperparameters for HTML and plain-text corpora.

Parameter	HTML	TXT	Description
<code>chunk_size</code>	1 000	500	Maximum characters per chunk
<code>chunk_overlap</code>	150	50	Overlap between consecutive chunks
<code>anchor_overlap</code>	150	–	Context prepended from previous section

Chunks are stored in ChromaDB in separate collections per $(embedding_model, context_type)$ pair, where `context_type` $\in \{\text{HTML}, \text{plain-text}\}$ distinguishes the two indexed formats of the same corpus (Section 3.4), so that differences in retrieval quality are attributable solely to the embedding model. New chunks are submitted in provider-aware batches to respect API rate and payload limits (OpenRouter: 25 per call; Google: 10; local and Azure: up to 5 000), deduplicated by identifier, and persisted; subsequent runs skip already-indexed chunks, making partial reruns safe.

2. **Retrieval.** Given query q , the pipeline computes $E(q)$ and retrieves the top- k chunks by similarity. For cosine distance ChromaDB’s native HNSW index is used; for Euclidean, Manhattan, and Chebyshev a brute-force scan over all stored embeddings is performed. The k initial chunks are then expanded by a depth-first traversal of the hyperlink graph: for each chunk, up to n_{links} outgoing links are followed recursively to depth d , and candidates reached via each link are reranked by the fixed distance to the link anchor text, retaining the top m per link. The final context window is the deduplicated union of initial and link-expanded chunks. The values I choose for retrieval hyperparameters $k, d, n_{\text{links}}, m$ and the distance used as similarity metric are given in Table 3.3.

Table 3.3. Retrieval hyperparameters used throughout the RAG evaluation.

Parameter	Default	Description
k	5	Chunks from initial similarity search
d	2	Maximum DFS depth for link following
n_{links}	3	Maximum outgoing links per chunk
m	3	Top chunks kept after per-link reranking
distance	cosine	Default similarity metric

The link-following mechanism introduces an asymmetry relevant to the robustness analysis: a query that retrieves at least one correct initial chunk can still reach the target passage through graph traversal even if its embedding has shifted; a query that retrieves no correct initial chunk propagates its error through all subsequent expansions. This makes the pipeline a sensitive testbed for studying how early retrieval failures cascade into answer degradation.

3. **Generation.** Retrieved chunks are concatenated and passed to Azure `gpt-4o-mini` under a fixed system prompt that prohibits reliance on parametric knowledge, instructing the model to answer solely from the retrieved text. Fixing the LLM to a single provider ensures that differences in BERTScore across configurations reflect changes in retrieved context, not in generation behaviour. Temperature is set to 0 for reproducibility purposes.

Dataset and Evaluation Metrics

The pipeline architecture described above is evaluated on a specific knowledge base and a targeted set of queries designed to probe both in-domain and out-of-domain robustness. The RAG evaluation uses the official Rulex Platform documentation as its knowledge base and a set of four queries divided into in-domain and out-of-domain categories.

Knowledge base. The knowledge base is the official Rulex Platform documentation, version 1.3.x, comprising 239 HTML files generated by Sphinx [56]. It was selected because it constitutes a realistic, technically dense knowledge base whose rich internal hyperlink structure makes it a particularly informative testbed for link-aware retrieval. The corpus spans three documentation domains:

- **Rulex Platform:** an overview of the platform as a data management system, introducing its two main components: Factory and Studio;
- **Rulex Factory:** the processing engine for building data flows, offering tools for import, transformation, optimisation, rule management, and scheduling;
- **Rulex Studio:** the front-end environment for creating interactive dashboards for data visualisation.

The corpus is indexed in two formats, each stored in a separate ChromaDB collection. The HTML export preserves the full hyperlinked structure and enables the depth-first graph traversal of Section 3.4; the plain-text version strips all markup and serves as a baseline that isolates pure vector similarity without link-following. The two formats use the chunking parameters appropriate to each (Table 3.2): the HTML pipeline splits at paragraph-anchor boundaries before character-level chunking, while the plain-text pipeline applies recursive character splitting alone. Within each format, the embedding model, generation LLM, system prompt, and decoding temperature are held fixed across all runs, so that any variation in retrieval or answer quality is attributable solely to the typographical perturbation of the query.

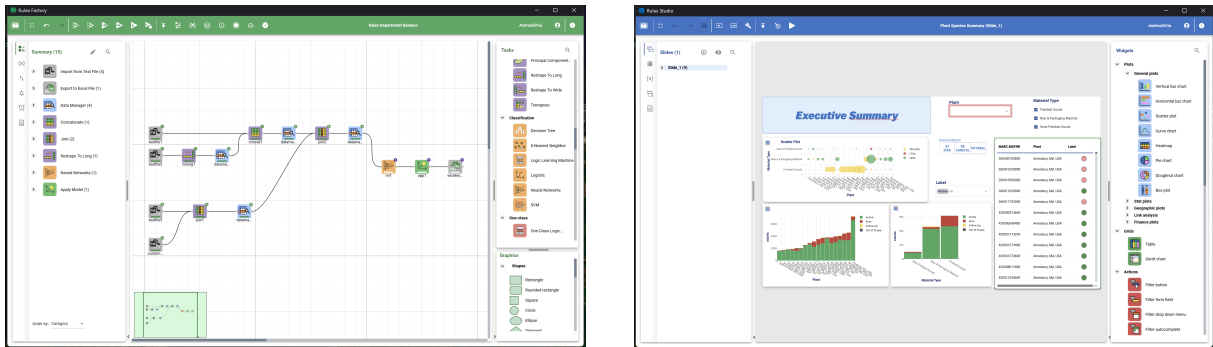


Figure 3.3. The left panel displays the dashboard of Rulex Factory, while the right panel shows that of Rulex Studio.

The knowledge base just described is queried through the set of inputs defined below.

Query set. Four queries divided into two categories are used for the RAG evaluation (Table 3.4), two per category. *In-domain* queries are factual questions whose correct answer exists in the Rulex Platform documentation; a well-functioning pipeline should retrieve relevant chunks and produce a grounded answer for all of them. *Out-of-domain* queries have no answer in the knowledge base: the pipeline should consistently return a *not found* response regardless of perturbation level. The reduced query set with respect to the embedding stability experiment (Section 3.3) reflects the higher per-call cost of the RAG pipeline, which requires a full LLM generation for each perturbed variant; $N_{\text{typo}} := 30$ independent perturbed queries are generated per (query, ρ) pair, a sample size sufficient to stabilise the Jaccard and BERTScore estimates while keeping the total number of LLM calls tractable. Queries 3 and 4 of Table 3.4 also appear in the embedding stability query set (Table 3.1, respectively entries 5 and 16), enabling a direct comparison between geometric displacement in the latent space and end-to-end retrieval degradation for the same input.

Table 3.4. Query set used in the RAG evaluation, grouped by category.

#	Category	Query
1	In-domain	what are the main components of rulex platform
2	In-domain	how do i filter data in rulex factory
3	Out-of-domain	what is the best way to handle a bank overflow
4	Out-of-domain	what is a fandango

The indices defined so far live entirely inside the embedding space: they quantify how far a vector moved, but not whether that movement had any consequence for the user. A displacement is harmful only if it changes what the system retrieves, and a change in what is retrieved is harmful only if it changes the answer that is finally produced. These are two distinct failure points, and they are measured by two distinct metrics. Their

definitions translate the geometric displacement framework of Section 3.3 into end-to-end system behaviour: one at the retrieval level, one at the generation level.

Retrieval stability: Jaccard overlap. Let $\mathcal{R}(q) \subseteq \mathcal{C}$ denote the set of chunks retrieved by the pipeline for query q , where $\mathcal{C}$ is the full chunk index. For each clean query q and each of its N perturbed variants $\tilde{q}^{(i)}$, the per-sample Jaccard overlap is

$$J_i := \frac{|\mathcal{R}(q) \cap \mathcal{R}(\tilde{q}^{(i)})|}{|\mathcal{R}(q) \cup \mathcal{R}(\tilde{q}^{(i)})|}, \quad (3.4.1)$$

and its average

$$\bar{J}(q, \rho) := N^{-1} \sum_i J_i \quad (3.4.2)$$

is reported. A value of $\bar{J} = 1$ means the perturbed query retrieves exactly the same chunks as the clean one; $\bar{J} = 0$ means complete divergence.

Jaccard answers the question: *is the system still looking at the same documents?* It compares two sets of chunk identifiers and nothing more, ignoring their content, their internal ranking, and their relevance to the query. This apparent crudeness is deliberate, and the reason lies in how retrieval actually behaves.

Top- k selection is discontinuous. As the perturbed query vector drifts away from the clean one, the retrieved set does not change at all, until the drift crosses a boundary between two regions of the embedding space, at which point one chunk is swapped for another in a single step. There is no intermediate state in which a document is partially retrieved. The consequence, and the reason this experiment exists, is that the relationship between geometric displacement and retrieval failure is not proportional: a small shift may be entirely harmless, and a shift only marginally larger may replace the retrieved set completely. A metric defined on similarity scores, being continuous, would average over these discontinuities and conceal precisely the behaviour under investigation; a set-overlap metric registers them directly. Jaccard is therefore the index to consult when the question is *whether the displacement measured by NED actually crossed a decision boundary*, and it is the natural bridge between the geometric analysis of Section 3.3 and the end-to-end evaluation that follows.

Results are reported separately for in-domain and out-of-domain categories and aggregated within each category by averaging $\bar{J}$ over queries at each budget ρ , with ± 1 standard deviation across queries shown as a shaded band.

Jaccard, however, cannot see the last step of the pipeline. A perturbed query may retrieve a completely different set of chunks and still yield a perfectly adequate answer, if the corpus contains the same information in more than one place, or if the LLM can reconstruct the answer from a partially overlapping context. Answer quality is therefore measured separately at the generation level.

Answer stability: BERTScore. At the generation level, answer quality is measured by BERTScore F_1 [64]. Given a fixed reference answer $\hat{y}$ (generated from the clean query,

3.5 Summary

This chapter has formalised the three experimental modules that constitute the empirical core of the thesis, grounding each one in the mathematical and computational framework developed in Chapters 1 and 2.

- **Section 3.1** stated the central research question in metric-space terms: given a clean query q and a perturbed variant $\tilde{q}$ at Damerau-Levenshtein distance δ , does the inequality $d_{\text{out}}(E(q), E(\tilde{q})) \leq L \cdot \delta$ hold for a finite constant L , and what is its empirical value across models and metrics? The stronger bi-Lipschitz requirement adds a lower bound controlled by c , giving the distortion index $\kappa := C/c$ as the primary scalar summary of embedding robustness. The Normalised Embedding Displacement (NED) is introduced as the pointwise ratio of output-space displacement to input-space edit distance, and the three nested research questions addressed by the thesis are made explicit: noise structure, embedding stability, and retrieval impact.
- **Section 3.2** instantiated the Markov-chain noise channel of Section 1.3 as a concrete experimental tool. The character-level confusion matrix Q is estimated from two complementary spelling corpora (Birkbeck and Wikipedia) via Needleman-Wunsch alignment and Laplace smoothing, combining three complementary perturbation mechanisms: QWERTY substitution, Markov substitution, and transposition. Three spectral diagnostics validate the fitted matrix before it is used as a noise source: the ergodicity analysis verifies irreducibility and aperiodicity, the stationary distribution π is compared against empirical English letter frequencies to detect corpus bias, and the confusability indices χ , $\tilde{\chi}$, and $\hat{\chi}$ identify which characters are disproportionately error-prone. The pooled corpus strategy is adopted as the primary noise source, with the averaged strategy retained as an ablation baseline.
- **Section 3.3** operationalised the bi-Lipschitz framework as a set of computable robustness indices. Perturbation intensity is parameterised by the relative DL budget ρ , which targets a fixed fraction of the query length and ensures cross-query comparability. For each of the six embedding models and four output-space distance functions, $N_{\text{typo}} := 500$ independent perturbed queries are generated per (query, ρ) pair and filtered to remove zero-DL and malformed samples. The resulting NED family (NED, U-NED, L-NED, and Distortion Index κ) constitutes the empirical bi-Lipschitz band of each model, while the displacement thresholds ρ_{50} , ρ_{75} , and ρ_{90} summarise the saturation profile of the displacement curve. All metrics are computed on provider-returned vectors without additional ℓ_2 -normalisation, so that cosine and Euclidean distances remain genuinely distinct measurements.
- **Section 3.4** extended the stability analysis to an end-to-end RAG pipeline built on LangChain and ChromaDB, using the official Rulex Platform documentation as its knowledge base. The same six embedding models serve as retrieval backbones,

with Azure `gpt-4o-mini` fixed as the generation model to isolate retrieval effects. Two metrics operationalise the three-level experimental hierarchy: Jaccard overlap measures retrieval stability at the set level, capturing the all-or-nothing behaviour of nearest-neighbour retrieval under embedding shift, while BERTScore F_1 measures generation quality at the semantic level, capturing how much of the reference answer survives typographical corruption of the query. The four evaluation queries, divided into in-domain and out-of-domain categories, include two entries shared with the embedding stability experiment, enabling a direct comparison between geometric displacement in the latent space and end-to-end system degradation.

4 Experimental Results

This chapter presents the empirical results of the three experimental modules introduced in Chapter 3, following the methodological hierarchy established there: noise characterisation, embedding stability, and RAG evaluation. The three sections are designed to be read in sequence, because the findings of each level directly motivate and constrain the interpretation of the next: the confusability structure of the noise channel predicts which queries will show the largest NED, the NED and distortion profiles predict which providers will be most resilient in the RAG pipeline, and the RAG results confirm, or complicate, the geometric picture. Where the two levels tell different stories, the discrepancy is itself the finding.

Each result reported below is accompanied by an account of its operational meaning, since a distortion index of 105 or a mean NED of 0.077 conveys nothing in isolation to a reader deciding which model to deploy. What conveys meaning is the answer to questions of the form *how many typographical errors can a user commit before this model ceases to function*, or *which queries is this model likely to fail on*. The indices constitute the evidence; the interpretation drawn from them is presented alongside, and not as a separate register.

The perturbation budget in concrete terms.

The relative budget ρ (Definition 1.3.14) is the single axis along which all experiments are parameterised, and its meaning is worth fixing at the outset in terms of characters actually typed. A budget ρ applied to a query of length ℓ targets $\lfloor \rho \cdot \ell \rfloor$ Damerau-Levenshtein edits, with the accept/reject sampler guaranteeing at least one edit even where the floor would be zero. Table 4.1 reports the edit counts actually realised on four of the probe queries, and Table 4.2 the corrupted strings they produce.

Table 4.1. Mean number of character edits actually realised at each budget, for four probe queries of increasing length. Values are the recorded `mean_dl` of the experiment. The reading in the last column is used as a mental scale throughout the chapter.

ρ	<i>fandango</i> (18 ch)	<i>gradient</i> (35 ch)	<i>bank overflow</i> (46 ch)	<i>stale bread</i> (64 ch)	Reading
0.01	1	1	1	1	one slip, anywhere
0.02	1	1	1	1	one slip, anywhere
0.05	1	2	2	3	a distracted typist
0.08	1	3	4	5	a hurried, uncorrected query
0.15	3	5	7	10	a badly mistyped query
0.25	4	9	12	16	barely recognisable text

Two consequences follow, and they account for most of the asymmetries reported later in the chapter. First, the same ρ does not represent the same experience for a short and a long query: at $\rho = 0.08$ the *fandango* query receives a single edit, which may well fall on the one character that carries its meaning, while the *stale-bread* query receives five, distributed across a sentence that remains readable. Second, the regime that matters in deployment is the low one. Real users commit one or two errors per query rather than sixteen, so the operationally relevant range is $\rho \in [0.01, 0.08]$; results at $\rho = 0.25$ are reported for completeness of the curve rather than as a realistic scenario.

Table 4.2. Perturbed variants actually produced by the noise channel, drawn from the experimental records. Most remain comprehensible to a human even at high budgets, which is precisely the point: the question of this thesis is whether they remain comprehensible to the *model*.

ρ	Edits	Perturbed variant
<i>Clean:</i> what is a fandango		
0.02	1	what is a fadnango
0.08	1	what i sa fandango
0.15	3	what isbabfandnago
0.25	4	what i amvanadngo
<i>Clean:</i> what is the best way to handle a bank overflow		
0.02	1	what is the best way to handle a banko verflow
0.08	4	what is the best aay o handle a bak oevrflow
0.15	7	what is thhe bset awy toh zndle a bnak overfolw
0.25	12	whzti ste estwa yto hadnle w bakn ovrflwo

The contrast between the two blocks of Table 4.2 is itself instructive. The first perturbation of “*what is a fandango*” at $\rho = 0.02$ produces *what i sa fandango*: not a single letter has been altered, and only a space has migrated one position to the right. To a human reader the query is unchanged. To the tokeniser it is a different string, and Table 4.4 will identify it as the most damaging query in the entire study.

Table 4.3 summarises the full experimental grid, ordered by the sequence in which each axis is discussed.

Table 4.3. Full experimental grid across all three modules. Axes are listed in the order they are discussed in the chapter.

Axis	Values	Card.
Corpus combination	Pooled (Birkbeck + Wikipedia)	–
Smoothing	Laplace, $\lambda = 10^{-6}$	–
Perturbation	character-level	1
DL budget ρ	0.01–0.10 (step 0.01), 0.15, 0.20, 0.25	13
Queries	20 probe queries (Table 3.1)	20
Samples per cell	$N_{\text{typo}} = 500$, $ \mathcal{V}_q = 500$ (filters inoperative)	–
Embedding models	Azure, Gemini, HF $\times 2$, Ollama, OR_nvidia	6
Distance functions	cosine, Euclidean, Manhattan, Chebyshev	4

4.1 Typographical Noise Characterisation

The noise model used throughout the experiments is the pooled, Laplace-smoothed Markov chain estimated from the combined Birkbeck and Wikipedia corpora (Section 3.2). Alignment pairs from both sources are concatenated before computing the count matrix, so that corpus size proportionally determines influence on the final distribution. Laplace smoothing with $\lambda = 10^{-6}$ is applied to each row to eliminate zero-probability transitions without materially altering the dominant entries. The pooled strategy weights error types by observed frequency, and Birkbeck’s larger volume ensures that cognitive spelling confusions dominate over the keyboard-mechanical errors captured by the dedicated QWERTY channel. The averaged strategy is retained as an ablation baseline; the two strategies produce qualitatively identical spectral diagnostics and the averaged results are not reported separately.

This first module is not itself a result about embedding models: it is the instrument with which those models are tested, and the present section establishes that the instrument is calibrated. It asks whether the errors generated are the errors humans actually commit, which characters are the damaging ones, and whether the noise source is statistically stable enough to be reproducible. Only once these are settled can any displacement measured downstream be attributed to the embedding model rather than to an artefact of the perturbation.

Character Transition Matrix and Transition Graph

The pooled, Laplace-smoothed transition matrix $Q \in [0, 1]^{28 \times 28}$ is displayed alongside its thresholded directed graph ($q_{ij} \geq 0.05$) in Figure 4.1. Rows index the source character and columns the typed outcome; ε is the deletion/insertion sentinel and occupies the

final column and row of the heatmap. The two representations are complementary: the heatmap reveals the magnitude of individual entries, while the graph highlights the hub structure of the strongest transitions.

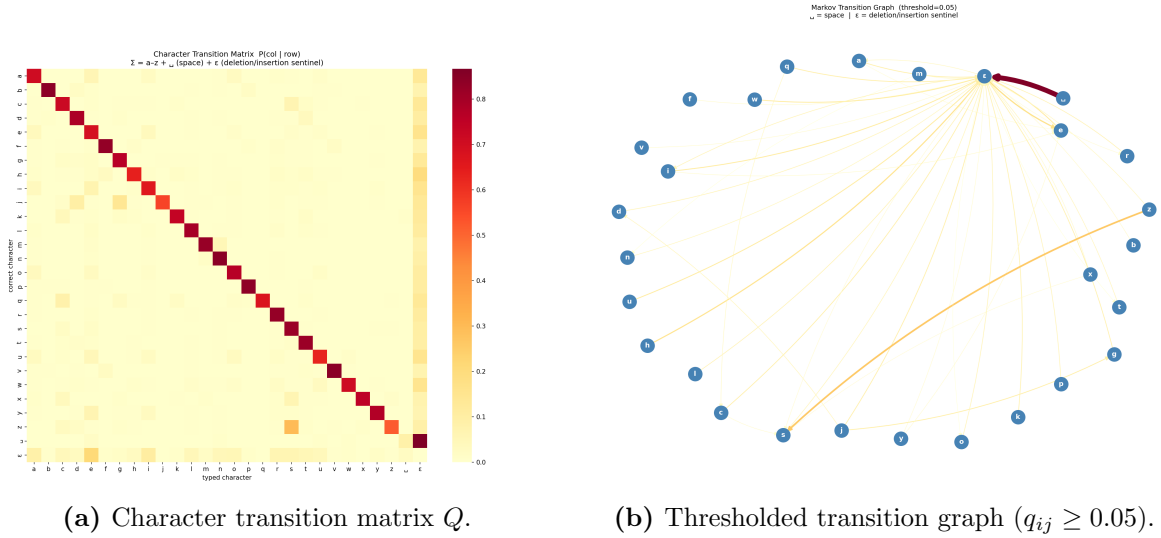


Figure 4.1. Pooled, Laplace-smoothed transition matrix Q and its thresholded directed graph ($\lambda = 10^{-6}$, pooled corpus).

The matrix exhibits strong diagonal dominance: q_{cc} is the largest entry in every row, confirming that characters are reproduced correctly the majority of the time. Diagonal entries range from 0.086 for $_$ to 0.846 for v . Off-diagonal mass is sparse and concentrates on phonetically motivated pairs: the strongest non-self, non-deletion arc is $z \rightarrow s$ ($q_{zs} = 0.299$, the orange arc in Figure 4.1b), driven by *realise* $\rightarrow$ *realize*-class sibilant substitutions. Other notable substitutions include $j \rightarrow g$ (0.140), $i \rightarrow e$ (0.071), $c \rightarrow s$ (0.065), and $m \rightarrow n$ (0.053), all reflecting phonological equivalence or orthographic proximity.

The gap symbol ϵ is the dominant hub: every character has an outgoing arc above threshold toward it, with the strongest being $_ \rightarrow \epsilon$ (dark-red, $q = 0.867$; space-deletion: *some times* $\rightarrow$ *sometimes*). In the reverse direction, ϵ is also an active source of insertions: its strongest outgoing arc leads to e ($q_{\epsilon e} = 0.212$), followed by i (0.106) and a (0.086), confirming that inserted characters preferentially realise as high-frequency vowels.

The raw confusability $\chi(c) := 1 - q_{cc}$ summarises per-character error probability directly from the diagonal. Among regular alphabetic characters values span [0.154, 0.479]. The five most confusable letters are z ($\chi = 0.479$), j (0.435), u (0.367), h (0.360), and i (0.338); the five least confusable are v (0.154), n (0.156), b (0.160), p (0.163), and f (0.168). The high $\chi(z)$ reflects both a sample-size effect (z is rare in English, so a modest off-diagonal count inflates the ratio) and a genuine phonological pattern via the $z \rightarrow s$ substitution. The high confusability of u , h , and i instead reflects genuine phonological ambiguity: u is frequently substituted by o (*colour* $\rightarrow$ *color*); h is regularly dropped in unstressed position (*which* $\rightarrow$ *wich*); i attracts vowel substitutions primarily toward e . The space character reaches $\chi(_) = 0.914$, driven almost entirely by space-deletion errors. These results are visualised in Figure 4.2.

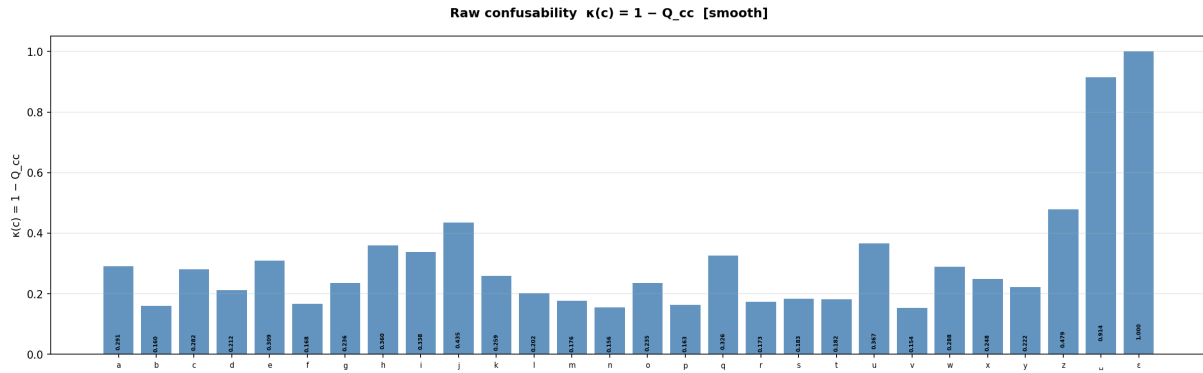


Figure 4.2. Raw confusability $\chi(c) := 1 - Q_{cc}$ (Laplace-smoothed, pooled corpus). **z** is the most confusable regular alphabetic character ($\chi = 0.479$), followed by **j** ($\chi = 0.435$); **v** is the least confusable ($\chi = 0.154$). The space character ($\chi = 0.914$) and gap symbol ($\chi = 1.000$) dominate as non-alphabetic entries.

Interpretation of the fitted channel.

Read as a description of human behaviour, the matrix is immediately recognisable. The most frequent error is not the mistyping of a letter at all: it is the loss or displacement of a space, which turns *some times* into *sometimes*. Beyond space loss, errors are predominantly phonetic rather than mechanical. Writers substitute letters that sound alike (**z**→**s**, **c**→**s**, **m**→**n**) or that reflect orthographic uncertainty (*colour/color*), and where a character is inserted in error it is almost invariably a vowel. The channel is governed by how words sound rather than by the arrangement of the keyboard.

This has a direct consequence for anyone reusing the noise model. Where the target population types on a physical keyboard from memory, as in the Birkbeck corpus, the present matrix is the appropriate one. Where the population types on a mobile device, and mechanical slips and autocorrection artefacts predominate, the QWERTY channel of Section 3.2 should be weighted more heavily relative to the Markov one, since the pooled matrix deliberately under-represents the mechanical component in order to avoid double-counting it.

Ergodicity and Stationary Distribution

The strongly connected component decomposition of the smoothed matrix confirms a single SCC spanning all 28 states. Laplace smoothing assigns positive probability to every transition and every state carries a self-loop ($Q_{cc} > 0$), so the chain is irreducible and aperiodic by construction, satisfying Theorem 1.3.5 and Corollary 1.3.5.

Figure 4.3 reports π alongside the rescaled English letter-frequency distribution [59]. The most closely aligned characters are **a** ($\pi_a = 0.078$ vs. English ≈ 0.082) and **n** ($\pi_n = 0.063$ vs. English ≈ 0.067). The three most over-represented characters are **e** ($\pi_e = 0.137$ vs. English ≈ 0.127), **s** ($\pi_s = 0.081$ vs. English ≈ 0.063), and **r** ($\pi_r = 0.070$ vs. English ≈ 0.060): all three attract a disproportionate share of substitution and deletion errors in

the corpus. The most notable under-representations are $\mathfrak{t}$ ($\pi_{\mathfrak{t}} = 0.060$ vs. English ≈ 0.091), $\mathfrak{o}$ ($\pi_{\mathfrak{o}} = 0.054$ vs. English ≈ 0.075), and $\mathfrak{d}$ ($\pi_{\mathfrak{d}} = 0.030$ vs. English ≈ 0.043). The gap symbol ε holds $\pi_{\varepsilon} = 0.110$ (the second-highest mass after $\mathfrak{e}$), reflecting the high aggregate deletion rate; the space character reaches near-zero stationary probability ($\pi_{\text{space}} \approx 0.004$) because its deletion rate of 87% causes the chain to exit that state almost immediately.

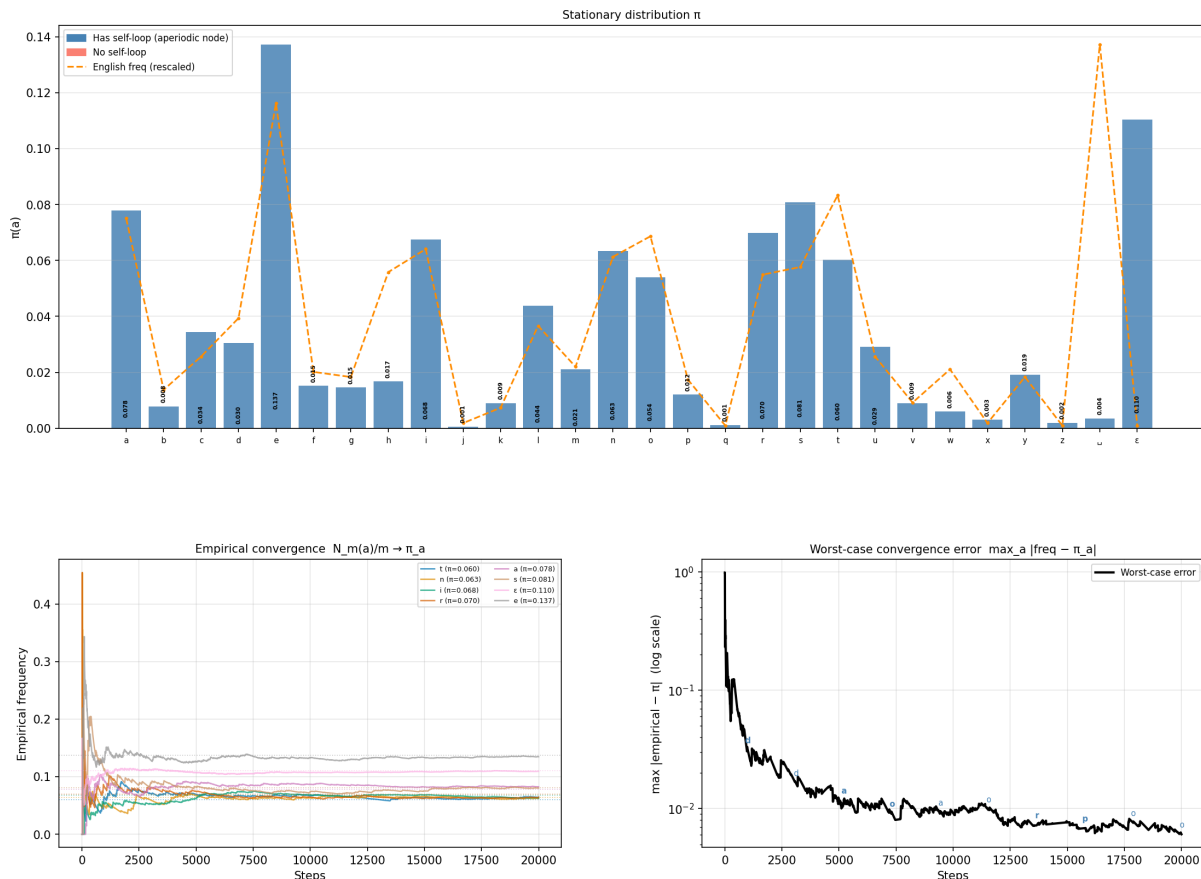


Figure 4.3. Top: stationary distribution π (blue bars) vs. rescaled English letter frequencies [59] (orange dashed line). Bottom-left: empirical occupation frequencies $N_m(a)/m$ for the eight highest-mass states over 2×10^4 steps. Bottom-right: worst-case deviation $\max_a |N_m(a)/m - \pi_a|$ (log scale); letter annotations identify which character achieves the maximum deviation at each step.

The Monte Carlo simulation (Section 3.2) confirms finite mixing time. All eight tracked occupation frequencies stabilise before $m = 5,000$ steps. The empirical mixing time is $t_{\text{mix}} \approx 3,000$ steps: the worst-case deviation Δ_m falls below 10^{-2} at that point and reaches $\approx 3 \times 10^{-3}$ at $m = 2 \times 10^4$, where the residual fluctuation is attributable to Monte Carlo variance. The character most persistently appearing as worst-case offender at later steps is $\mathfrak{o}$: since $\pi_{\mathfrak{o}} = 0.054$ lies well below its English frequency of ≈ 0.075 , the chain requires more steps to depress $\mathfrak{o}$'s visit frequency to its low stationary value than it does for characters whose π_c is closer to their natural-text baseline. The finite t_{mix} confirms that π is a genuine attractor of the channel and that the stationary distribution faithfully describes the asymptotic confusion pattern of the noise model.

Interpretation and scope of the mixing time.

Stripped of its formalism, the stationary distribution poses a sanity question about the instrument: if a text were corrupted repeatedly, what would the resulting distribution of characters look like? Were the answer markedly unlike English, the channel would be driving every query toward the same degenerate string irrespective of what had been typed, and any displacement measured downstream would be an artefact of the perturbation rather than a property of the model. It is not: Figure 4.3 shows the asymptotic frequencies tracking English closely, with deviations that are themselves interpretable (**e**, **s** and **r** attract errors and therefore appear more often than in clean text; deletions accumulate on ε).

Two clarifications are in order for the reader inclined to over-interpret t_{mix} . It does not imply that a query must be corrupted three thousand times before the noise model is valid: the perturbed queries used in the experiments are drawn independently, each generated afresh from the clean string, and their statistical properties do not depend on t_{mix} at all (Section 3.2). Its value is one of reproducibility: it certifies that the noise source possesses stable, well-defined statistics, so that the experiments of this chapter may be repeated and the same distribution of errors recovered.

Confusability Analysis

The two normalised confusability indices $\tilde{\chi}$ and $\hat{\chi}$ defined in Definition 3.2.1 are shown in Figure 4.4. They reveal structure that the raw index obscures by separating two distinct sources of high confusability: distributional rarity in the corpus versus intrinsic phonetic or keyboard difficulty.

Under the stationary-normalised index $\tilde{\chi}(c) := \chi(c)/\pi_c$, **j** is the dominant outlier ($z = +4.56$): its near-zero stationary mass amplifies even a moderate raw χ . A secondary outlier is **q** ($z \approx +1.29$), which shares the same sample-rarity mechanism. All other alphabetic characters cluster between $z = -0.4$ and $z = +0.1$, indicating that their confusability is broadly proportional to their mutation-target frequency.

Under the frequency-normalised index $\hat{\chi}(c) := \chi(c)/f(c)$, ε appears as an extreme outlier; however, since ε does not appear in natural text, $f(\varepsilon) \rightarrow 0$ by construction and $\hat{\chi}(\varepsilon)$ is uninformative. Among alphabetic characters, the principal outliers are **z** ($z \approx +1.91$) and **q** ($z \approx +1.17$): both are rare in English yet accumulate disproportionately high substitution counts.

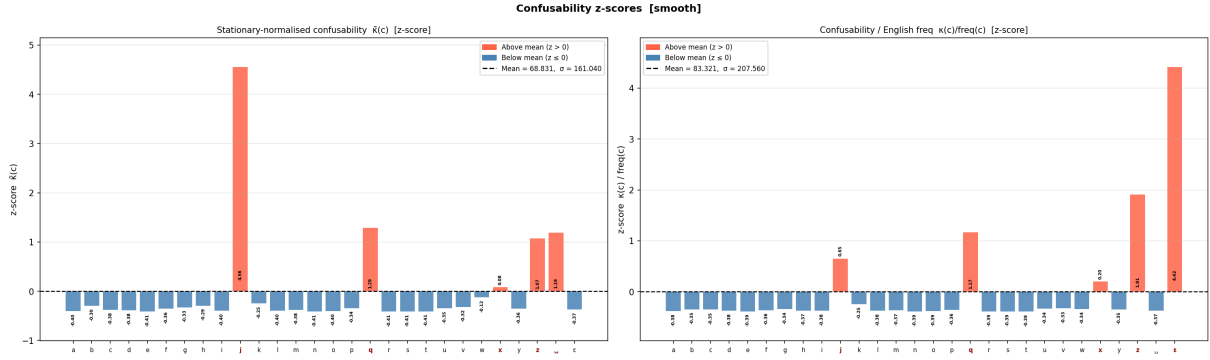


Figure 4.4. Normalised confusability z-scores (pooled corpus, smoothed). Left: $\tilde{\chi}(c) := \chi(c)/\pi_c$; j is the dominant outlier ($z = +4.56$); q is a secondary outlier ($z \approx +1.29$). Right: $\hat{\chi}(c) := \chi(c)/f(c)$; ε dominates as a modelling artefact (see text); z is the main alphabetic outlier ($z \approx +1.91$).

The characters that drive embedding displacement at low noise levels satisfy two joint conditions: moderate-to-high χ *and* high occurrence in the probe queries. Characters such as j , q , and z carry the highest raw confusability ($\chi \in [0.33, 0.48]$) but appear collectively only a handful of times across the 20 queries of Table 3.1, contributing negligible cumulative noise. The dominant argument for the full three-channel perturbation model is straightforward: any character that appears frequently has proportionally more chances of being affected by any of the three operations per perturbation step (Section 3.2).

The five characters with the highest product of χ and query-set count are e ($\chi = 0.309$, 84 occurrences across 18 of 20 queries), t ($\chi = 0.183$, 73 occurrences in all 20 queries), h ($\chi = 0.360$, 55 occurrences in all 20 queries), o ($\chi = 0.235$, 58 occurrences in all 20 queries), and a ($\chi = 0.291$, 52 occurrences in 17 of 20 queries). Their cumulative contribution to both DL distance and embedding displacement dominates over any single high- χ rare character such as z or j , which appear at most a handful of times across the query set. Two specific predictions follow for the embedding stability results of Section 4.2: short queries containing a high density of high- χ characters are expected to show larger NED at small ρ , because even a single edit is likely to land on a semantically loaded, highly error-prone character; and the dominance of e , t , h , and o across the query set means that the aggregate noise signal is concentrated in a narrow band of the alphabetic space, which should produce coherent geometric trajectories in the t-SNE projections of Section 4.2.

Confusability versus impact.

The two normalised indices answer questions that are easily conflated. Which letters are most often mistyped is answered by the raw χ , and the answer is z , at nearly 50%. Which letters are mistyped disproportionately often relative to their frequency in natural text is answered by $\hat{\chi}$, and yields the same candidates, z and q .

Neither, however, is the question that matters for the present purpose. A letter can damage a query only if it occurs in that query, and z , q and j together appear a handful

of times across all twenty probe queries. The letters that in fact break a query are the unremarkable ones: **e**, **t**, **h**, **o**, **a**. Each is only moderately error-prone in isolation, but their ubiquity ensures that almost every edit the channel produces lands on one of them.

The prediction that follows is sharp, and is confirmed quantitatively in Section 4.2. The most fragile query in the set should be “*what is a fandango*”. Of its eighteen characters, only nine belong to **fandango**, the sole content word: **what**, **is** and **a** are function words carrying almost no retrieval signal, so the semantic weight of the query rests on a single token. Two failure routes follow. An edit landing within **fandango** corrupts the only word identifying what is being asked; an edit landing on a space, the most probable single event in the entire channel ($\chi(_) = 0.914$), re-segments the string and may cause the tokeniser to split **fandango** differently even where its letters survive intact, as in the recorded variant **what i sa fandango**. It proves, as anticipated, to be the worst-case query for every one of the six providers. The rule of thumb that emerges is therefore a counter-intuitive one: the risk lies with the vowels, the common consonants and above all the spaces, not with the rare letters, and short queries whose meaning rests on one or two content words are those most liable to break.

4.2 Embedding Stability Results

This section reports the empirical results of the embedding stability module described in Section 3.3. The presentation is organised to build the picture from the most immediately interpretable view to the most detailed: the embedding-vs-DL comparison introduces the dominant phenomenon (front-loaded saturation), the displacement thresholds quantify when it occurs, the bi-Lipschitz band characterises how it varies across providers and metrics, the distortion index captures how selectively it affects individual queries, and the t-SNE projections give it a geometric interpretation.

The question underlying all of this is not itself geometric. It is whether, when a user commits a typographical error, the model still recognises what was meant. An embedding model answers that question by placing the query somewhere in a high-dimensional space, and to recognise what was meant is to place the corrupted query close enough to where the clean one would have gone. What follows measures how far the model moves the query, and whether it moves every query by a comparable amount or singles out a few for catastrophic treatment.

The six providers listed in Table 2.1 were evaluated on the complete query set of 20 queries across all 13 rate levels. Both filtering steps described in Section 3.3 remained inactive throughout: the accept/reject sampler achieved the target DL distance within 200 candidates for every (q, ρ) pair and no provider returned malformed embeddings, so $|\mathcal{V}_q| = N_{\text{typo}} = 500$ holds uniformly for all records.

Input-Output Distance Comparison

The most direct view of embedding stability is obtained by comparing the normalised embedding distance and the normalised Damerau-Levenshtein (DL) distance across ρ . For each query q , both quantities are divided by their respective per-query maxima over the tested range, mapping them to $[0, 1]$. The normalised DL distance is a straight diagonal by construction (linear in ρ); the normalised embedding curve is concave. Panels where the embedding curve lies above the diagonal are shaded **red**: the embedding has already accumulated a disproportionate fraction of its total excursion relative to the edit budget. Panels where the curve lies below are **green**: the embedding is still growing roughly proportionally to the number of edits.

The colour convention admits a simple reading. The dashed diagonal represents proportionate behaviour, in which two errors displace the query twice as far as one; the solid curve represents what the model in fact does. Red indicates over-reaction, the model having already carried the query most of the way toward its eventual position after only one or two errors; green indicates degradation in proportion to the quality of the typing.

Figure 4.5–4.7 show all six providers for each of the three qualitatively distinct metrics: cosine (angular displacement), Euclidean (ℓ_2), and Chebyshev (ℓ_∞). Euclidean and Manhattan produce near-identical patterns; Manhattan is reported in Appendix with the full per-query grids.

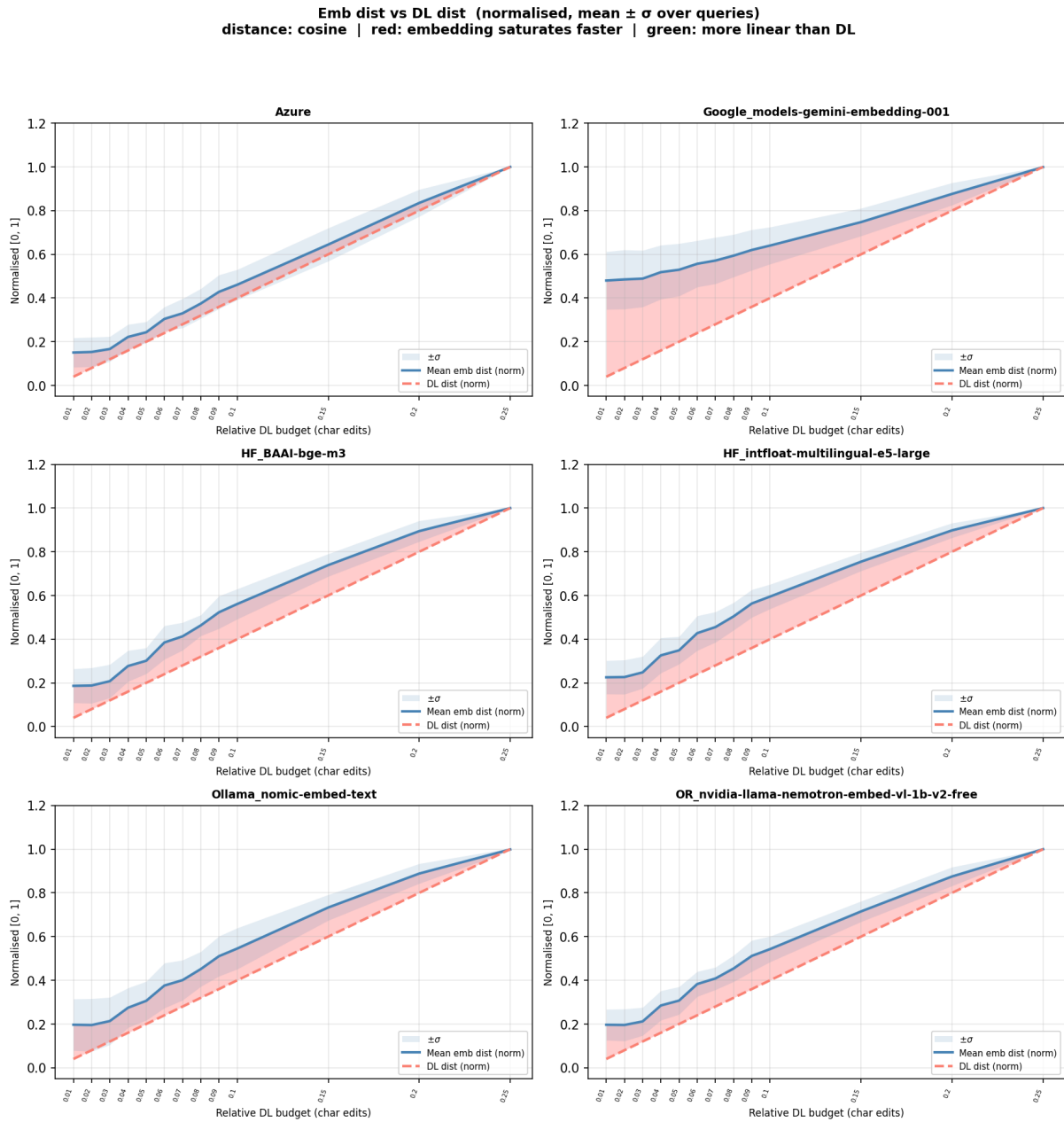


Figure 4.5. Mean normalised embedding distance vs. normalised DL distance under **cosine** similarity. Each panel shows one provider; the solid curve is the mean over 20 queries; the shaded band is $\pm \sigma$ std. **Red:** embedding saturates faster than the DL budget. **Green:** embedding grows proportionally to edit distance (not visible in this study case, more in Appendix).

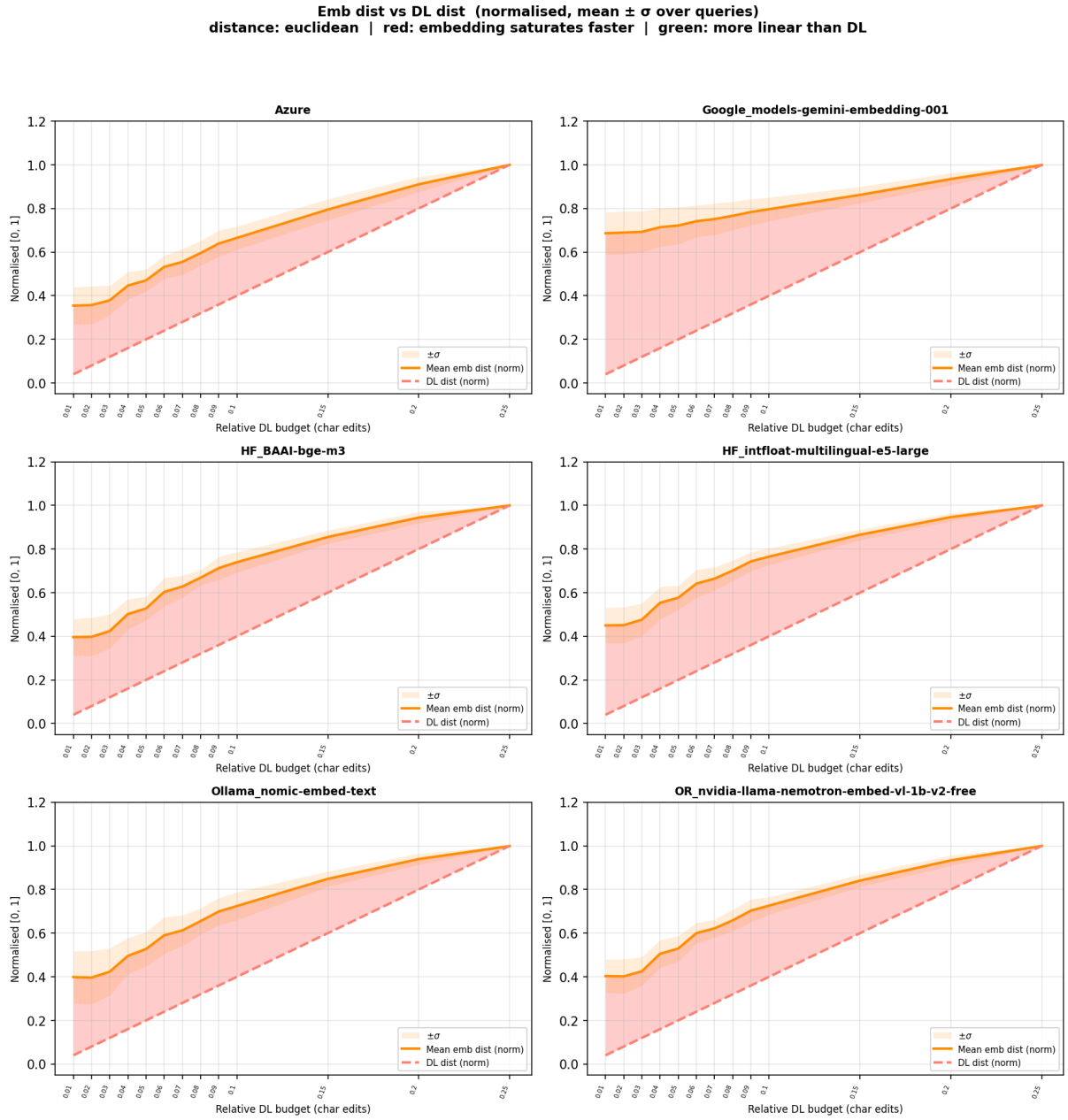


Figure 4.6. Mean normalised embedding distance vs. normalised DL distance under the Euclidean (ℓ_2) metric. Same layout and colour convention as Figure 4.5.

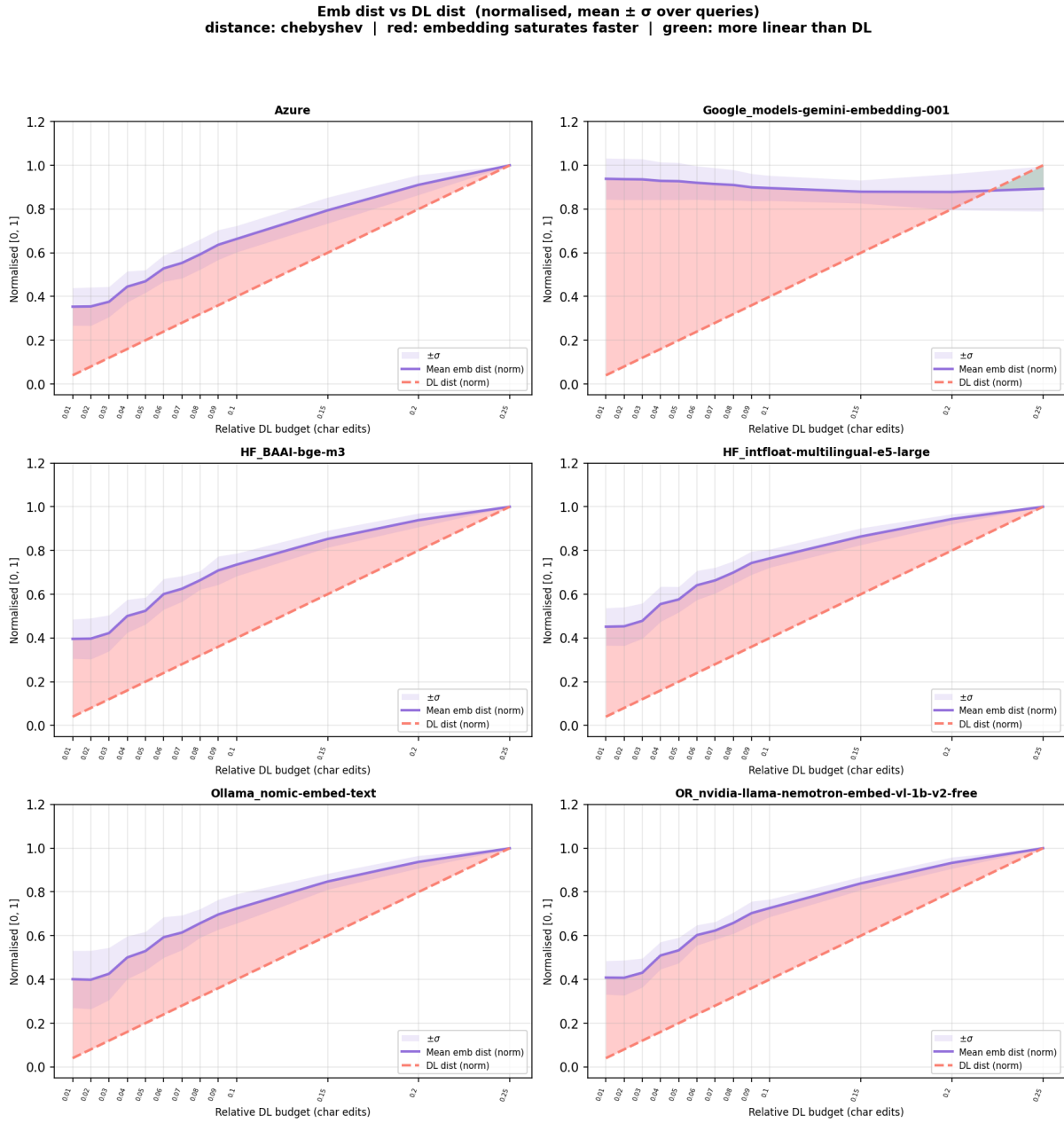


Figure 4.7. Mean normalised embedding distance vs. normalised DL distance under the Chebyshev (ℓ_∞) metric. Same layout as Figure 4.5.

Three observations emerge from the three figures.

Early saturation of the displacement curve.

Across all providers and all metrics, the red region dominates: the embedding curve rises steeply in the first few budget steps and then levels off while the DL denominator continues to grow linearly. A closer inspection of the per-provider per-query results, reported in Appendix and illustrated for three representative queries in Figure 4.8, reveals that Azure is the only partial exception under cosine distance: 5 of its 20 queries remain

in the green zone at $\rho = 0.05$, where the embedding grows no faster than the edit count, reflecting the slower, less sharply concave saturation visible in the Azure panel. For all other providers, the red zone covers roughly 90% of the tested budget range under cosine, and approaches 100% under Euclidean and Chebyshev.

The extent of this green region depends systematically on query length. For the longest queries the normalised embedding curve sits almost exactly on the DL diagonal: the 63-character “*what can you do with a degree in mathematics besides suffering?*” tracks it closely, and the 46-character “*what is the best way to handle a bank overflow*” runs marginally below it, in the green zone. Shorter queries sit further into the red zone, the 18-character “*what is a fandango*” being the clearest case. The mechanism is that at a fixed relative budget ρ a longer query receives proportionally more character edits, so the perturbation is spread across many tokens and no single corrupted character dominates the displacement; in a short query the same relative budget concentrates on one or two characters, producing a large, front-loaded angular jump. The trend is consistent rather than absolute, with the longest queries the most tightly aligned to the diagonal: cosine stays close to the DL diagonal at every query length, but the agreement is tightest for the longest queries.

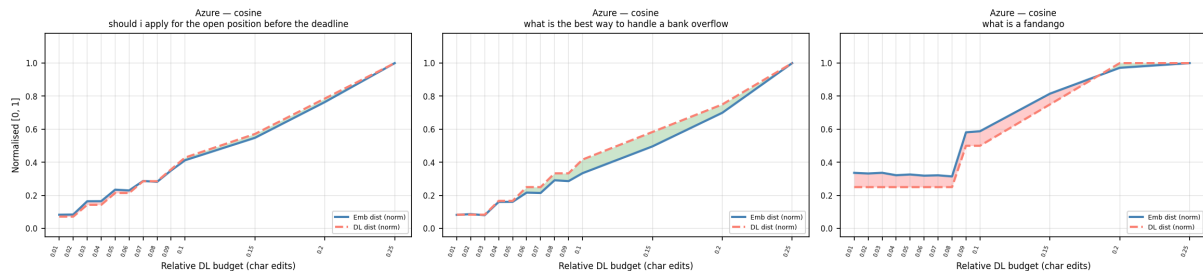


Figure 4.8. Embedding distance vs. DL distance under cosine for three representative Azure queries (normalised to $[0, 1]$ by their per-query maxima). The solid curve is the mean embedding distance, the dashed curve the linear DL distance. **Green** marks the budgets where the embedding grows no faster than the edit count, **red** where it saturates faster. The amount of red increases as the query gets shorter (right to left). The full per-query grid for all providers and metrics is in Appendix .

This is the most consequential finding of the embedding module, and it admits a blunt formulation: most of the damage a typographical error can do is done by the first one. A user who mistypes a single character has already carried the query most of the way toward the position a thoroughly corrupted version would occupy, and the second, third and fourth errors contribute comparatively little. The perturbation records render this concrete. Of `what is the best way to handle a banko verflow` (one edit) and `whzti ste estwa yto hadnle w bakn ovrflwo` (twelve edits), the second is barely readable while the first is unremarkable to a human eye; yet in the latent space the first has already traversed a large fraction of the distance covered by the second.

The implication runs counter to intuition. Effort devoted to handling severely corrupted queries yields poor returns, since by that point the model has long since lost the thread; what pays is the interception of the single error, and a spell-checking or

normalisation layer placed before the retriever, however rudimentary, addresses the regime in which nearly all of the harm is inflicted.

A second consequence concerns query length. A single misplaced character in the eighteen-character fandango query is catastrophic, whereas the same error in the sixty-four-character stale-bread query is inconsequential. Any system whose users type terse search-box queries is therefore materially more exposed to typographical noise than one whose users type complete questions, and should budget for query correction accordingly.

The role of metric geometry.

Under cosine (Figure 4.5) the gap between the two curves is narrowest: angular displacement accumulates more gradually and more linearly with the edit budget than ℓ_p displacement, because a perturbation that shifts the vector proportionally in all dimensions leaves the angle approximately unchanged. Under Euclidean (Figure 4.6) the red bands are wider, and under Chebyshev (Figure 4.7) the onset is earliest and steepest: a single large coordinate excursion saturates $\|\cdot\|_\infty$ even at $\rho = 0.02$.

The four distances are, in effect, four ways of asking how far the vector has moved, and they disagree because each is sensitive to a different kind of movement. Cosine registers only direction: a perturbation that nudges every coordinate slightly leaves the vector pointing essentially the same way, and cosine reports almost nothing. This makes it the most forgiving of small, diffuse displacements, which is precisely what a typographical error produces. Chebyshev registers only the single worst coordinate, so that one anomalous dimension saturates it even where the remaining three thousand are untouched, and it is correspondingly the most easily disturbed by an isolated excursion. Euclidean and Manhattan lie between, aggregating all coordinate deviations, and produce near-identical curves throughout, which is why only Euclidean is reported in the main text.

The choice of distance is therefore not a matter of convention, and it accounts for the standing of cosine as the default retrieval metric. Section 4.3 will show that replacing it with ℓ_∞ suffices on its own to destroy a working retrieval system: a metric that responds to a single wayward coordinate will reorder the nearest neighbours at the slightest provocation.

The anomalous behaviour of Gemini.

Gemini departs from the general pattern in a metric-dependent way, markedly under cosine and through an entirely different mechanism under Chebyshev.

Under cosine: anomalously early saturation.

Its cosine panel shows the steepest initial rise of all providers: by $\rho = 0.02$, the embedding has already accumulated a median of 48% of its total angular displacement, compared to 21% for HF_e5 at the same budget. The consequence is most clearly read through the mean NED $\hat{L}(\rho)$, the average ratio between embedding displacement and edit distance (Section 4.2). Because the angular displacement is front-loaded, it grows only marginally after the first few edits while the DL denominator keeps increasing linearly, so $\hat{L}$ decreases

sharply with ρ : the cosine mean NED of Gemini falls from the highest value of all providers at $\rho = 0.01$ ($\hat{L} = 0.125$) to the second lowest at $\rho = 0.25$ ($\hat{L} = 0.028$, behind only HF_e5). In other words, a perturbed Gemini query lands a large fixed angular step away from the clean query almost immediately, and additional edits move it comparatively little further.

Under Chebyshev: a flat, saturated excursion.

The behaviour of Gemini under Chebyshev is governed by a different mechanism, shown for the same three queries in Figure 4.9. Here the absolute displacement is essentially flat: the largest coordinate deviation barely changes across the whole budget range and is among the smallest of all providers ($C_{\text{cheb}} = 0.088$ at $\rho = 0.08$, second only to HF_e5). Because the ℓ_∞ displacement is both small and stable, the normalised curve sits near its maximum from the first budget in every panel and the DL diagonal does not catch up until $\rho = 0.25$, with $\rho_{50}^{\text{cheb}} = 0.01$, the earliest of all providers and metrics. Unlike the cosine case, query length has essentially no effect here (the corresponding length correlation is approximately 0), because the worst-case single-coordinate deviation that ℓ_∞ measures is already near its ceiling after one edit and stays there regardless of how the remaining budget is distributed. This is not a sign of fragility but of robustness under ℓ_∞ : no single coordinate of the Gemini embedding ever moves much. The contrast with the cosine case is therefore not a true reversal of behaviour but a consequence of what each metric measures: cosine captures a large but front-loaded angular rotation, whereas Chebyshev captures a small, saturated, per-coordinate excursion.

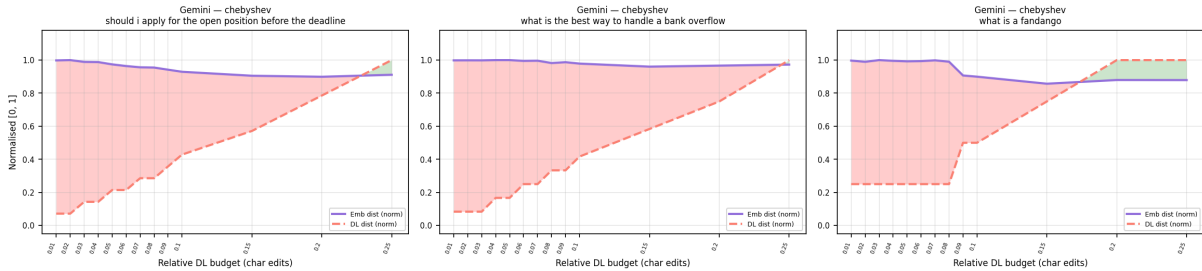


Figure 4.9. Embedding distance vs. DL distance under Chebyshev for the same three queries, this time for Gemini. The normalised embedding curve sits near its maximum from the first budget for every query, so the DL diagonal catches up only near $\rho = 0.25$: the ℓ_∞ displacement is essentially saturated at a single edit, independently of query length.

Stripped of the metric machinery, the behaviour of Gemini may be described as follows. The model responds to a typographical error by rotating the query vector, swinging it a fixed amount away from the clean position almost immediately and then largely ceasing to move, however much further the typing deteriorates; what it does not do is stretch any individual coordinate, so that internally very little changes.

Gemini is thus a model with a low tolerance for the first error and a high tolerance for everything subsequent, treating one error and five as nearly the same event. This is a liability where a deployment must distinguish a lightly mistyped query from a clean one, and an asset where a deployment must survive users who type poorly and consistently, since performance degrades early and then plateaus rather than collapsing further as

noise accumulates. It is accordingly the appropriate choice where predictability outweighs sensitivity, as in a system serving a population with known poor input quality, and the wrong choice where a clean query must reliably be told apart from a slightly corrupted one.

Displacement Thresholds

To quantify the saturation timing, I compute the displacement thresholds ρ_{50} and ρ_{75} (Definition 3.3.1): the first budget at which the embedding has accumulated 50% and 75% of its total observed displacement, respectively. The threshold ρ_{90} is not reported as a primary diagnostic because, with the tested range capped at $\rho = 0.25$ and the curve monotone by construction, it is always close to the upper boundary and does not reflect the true asymptote of the displacement curve; ρ_{50} is the most informative threshold because it is reached in the rapidly-growing portion of the curve where provider differences are sharpest. Figure 4.10 reports both thresholds grouped by provider.

Stated in one sentence, ρ_{50} is the number of errors, expressed as a fraction of query length, at which the model has already suffered half of all the harm that typing can inflict upon it. A low ρ_{50} denotes a model that gives way early; a high one, a model that degrades gradually. On a forty-character query, $\rho_{50} = 0.03$ corresponds to roughly one error, and $\rho_{50} = 0.15$ to roughly six.

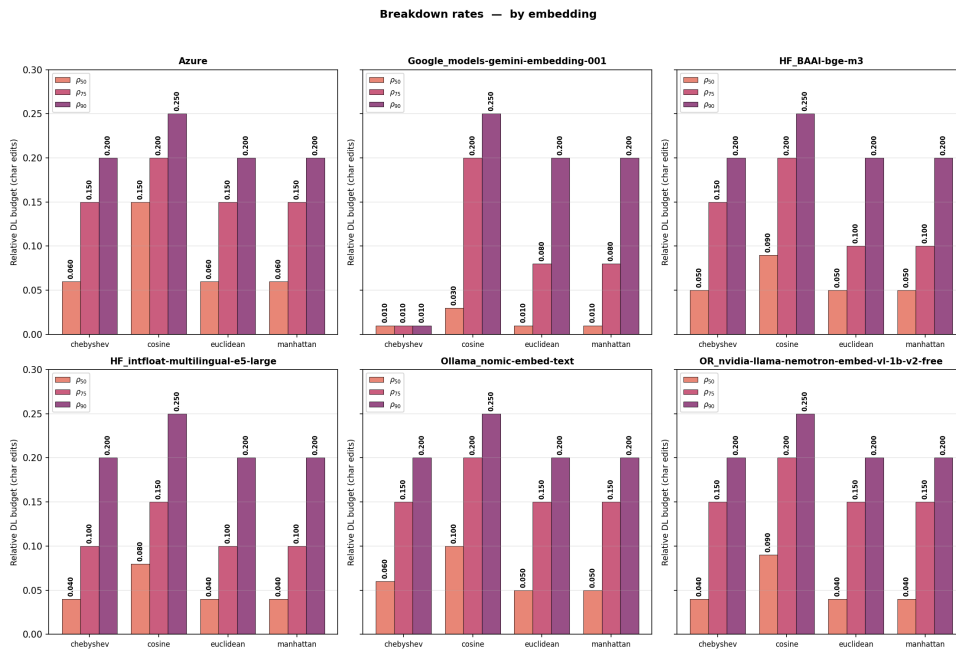


Figure 4.10. Displacement thresholds ρ_{50} and ρ_{75} (median over queries) grouped by embedding provider. Each cluster of bars corresponds to one distance function. Lower values indicate earlier saturation.

Cross-provider spread under cosine.

ρ_{50} ranges from 0.030 (Gemini) to 0.08 (HF_e5), 0.09 (HF_bge and OR_nvidia), 0.10 (Ollama), and 0.15 (Azure). The unusually low cosine value of Gemini confirms that half of its total angular displacement is absorbed within the first three budget steps. The late $\rho_{50} = 0.15$ of Azure does not indicate greater robustness: the curve is less concave because Azure starts with a higher mean NED at low budgets and accumulates displacement more gradually.

Translated onto a query of typical search-box length, around forty characters, the ranking becomes directly actionable. Gemini absorbs half its damage after roughly one character, and thus has essentially no tolerance for the first error. HF_e5, HF_bge, OR_nvidia and Ollama cluster together at three to four characters, absorbing a single error without much reaction. Azure requires six characters to reach its half-way mark and is thus the model that degrades most gradually as typing quality falls.

One caveat the ranking conceals deserves emphasis: a late ρ_{50} is not automatically a better one. Azure reaches its half-way mark late because it begins to move early and continues to move steadily, not because it ignores the first errors. The threshold measures the shape of the degradation curve rather than its height, and to know whether the displacement is large or small in absolute terms one must consult the mean NED of the following subsection.

Convergence under the ℓ_p norms.

Under Euclidean the picture is similar but compressed: ρ_{50} ranges from 0.01 (Gemini) to 0.04 (HF_e5, OR_nvidia) and 0.06 (Azure), with all providers converging within a narrower band than under cosine. Under Chebyshev, Euclidean, and Manhattan, all six providers converge to $\rho_{50} \in [0.04, 0.06]$. This confirms that the cross-provider spread in saturation timing is a primarily angular phenomenon, driven by angular rather than magnitude displacement. Beyond ρ_{75} , all providers reach 0.20 under the three ℓ_p metrics, and 0.20–0.25 under cosine, suggesting that provider-specific differences are most meaningful in the moderate regime $\rho \in [0.05, 0.15]$.

The consequence of this convergence is that the choice of provider matters only where retrieval is performed under cosine distance. Under the ℓ_p norms all six models behave nearly identically, so that a system retrieving with Euclidean distance derives almost no benefit from a premium embedding model: the metric has flattened away the very distinctions for which one is paying. This is the first of several results pointing in the same direction, namely that the distance function is a more consequential decision than the model.

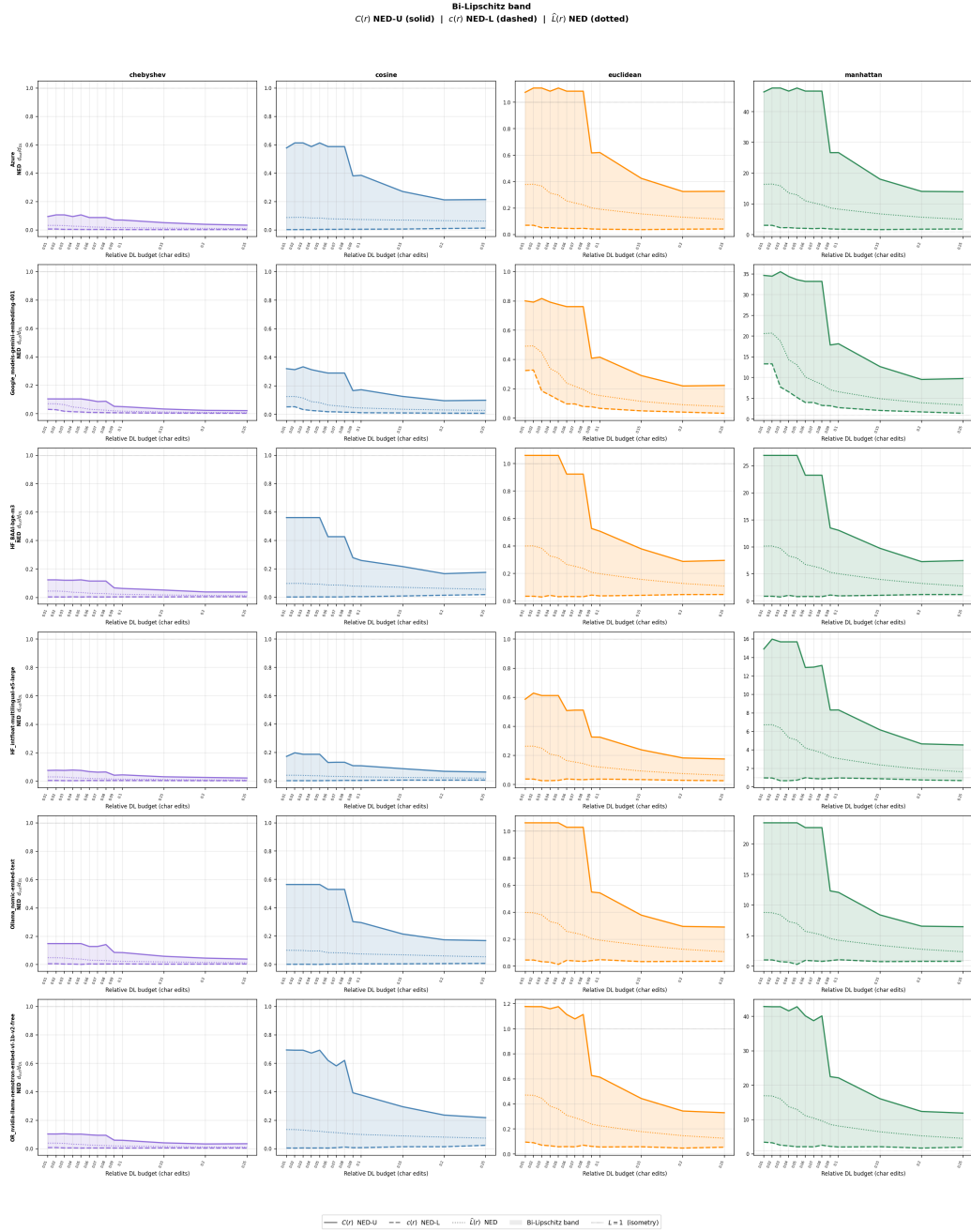


Figure 4.11. Empirical bi-Lipschitz band for all providers. Rows correspond to embedding providers; columns to the four distance metrics. Solid curve: U-NED $C(\rho)$; dashed: L-NED $c(\rho)$; dotted: mean NED $\hat{L}(\rho)$; shaded region: band $(c(\rho), C(\rho))$.

Bi-Lipschitz Band and Mean NED

With the saturation shape established, I now measure it quantitatively through the NED family. For each perturbed sample $\tilde{q}^{(i)}$, the pointwise ratio $\hat{L}_i := d_{\text{out}}(E(q), E(\tilde{q}^{(i)})) / d_{\text{DL}}(q, \tilde{q}^{(i)})$ is computed; aggregating within and across queries at each ρ yields the empirical bi-Lipschitz band $(c(\rho), C(\rho))$ and mean NED $\hat{L}(\rho)$,

plotted in Figure 4.11.

The band admits a direct reading. Its upper edge corresponds to the most unfavourably treated query, that on which a typographical error caused the greatest damage; its lower edge to the most favourably treated, where an error caused almost none. The dotted line between them is the average query. A model whose edges lie close together treats all its users alike, whereas one whose edges lie far apart behaves excellently on most queries and disastrously on a few.

Mean NED and the saturation drop.

Because d_{DL} grows linearly while d_{out} saturates, mean NED necessarily decreases with ρ . Under cosine, $\hat{L}$ drops by 28% ($0.087 \rightarrow 0.063$) for Azure, 77% ($0.125 \rightarrow 0.028$) for Gemini, 43% ($0.096 \rightarrow 0.055$) for HF_bge, 52% ($0.039 \rightarrow 0.019$) for HF_e5, 46% ($0.100 \rightarrow 0.054$) for Ollama, and 45% ($0.135 \rightarrow 0.074$) for OR_nvidia, mirroring the saturation patterns observed in Figure 4.5.

A natural misreading must be pre-empted. A falling mean NED does not indicate that the model improves as typing deteriorates. NED is a displacement per edit, and it falls because the denominator, the number of errors, continues to grow while the numerator, the damage, has ceased to. The model is not recovering; it has simply already done everything it was going to do. The steeper the fall, the more front-loaded the harm, which is why Gemini, at 77%, reacts most violently to the first error, and why Azure, at 28%, accumulates its damage most evenly.

Provider ranking in the moderate-noise regime.

The dotted mean-NED curves of Figure 4.11 also fix the relative standing of the providers in the moderate-noise regime. At the reference budget $\rho = 0.08$ the cosine mean NED orders the providers as HF_e5 ($\hat{L} = 0.031$), Gemini (0.055), Azure (0.077), Ollama (0.081), HF_bge (0.083), and OR_nvidia (0.108). This ranking measures *average* displacement per edit only; it says nothing about how evenly that displacement is spread across queries, which is the role of the distortion index $\kappa = C/c$ analysed in the next subsection. The two views need not agree, and indeed they do not: a provider can have a low mean NED here yet a high κ there, and the comparison between the two is one of the main findings of this section.

The ranking answers the question of how much a given model is displaced by a typical error committed by a typical user. HF_e5 is displaced least, by a factor of three and a half over the worst performer; OR_nvidia most, and is therefore the model with the highest baseline sensitivity to noisy input, to be paired with a query-correction layer or reserved for pipelines whose input is machine-generated and clean. Were a single number to determine the choice of model, and no other information available, this would be that number, and HF_e5 the selection.

No single number should determine it, however, for the reason developed in the following subsection. The mean is an average over twenty queries and says nothing about whether the model behaved reasonably on all twenty or excellently on nineteen and catastrophically on one. In a deployed system, the query treated catastrophically is

not an outlier to be averaged away; it is a user whose search returned nothing.

Interpretation and scope of the band width.

The width and the absolute scale of the bands must be read with the same care. Chebyshev bands appear visually compressed relative to Euclidean and Manhattan, but this is a pure scale effect: Chebyshev records only the single largest coordinate deviation, so its values are an order of magnitude smaller in absolute terms than the ℓ_2 ones, and a band that looks thin on the axis is not necessarily a band of low distortion. What determines distortion is the multiplicative ratio C/c between the upper and lower NED, not the absolute vertical gap $C - c$: a Chebyshev band running between $c = 0.004$ and $C = 0.088$ is numerically narrow yet has $\kappa \approx 22$, whereas a Euclidean band ten times wider in absolute terms can have a far smaller ratio. The bands should therefore always be compared through κ , never by eye.

The same caution applies to any comparison of embedding models across providers. Manhattan distances in this study reach values above 40 while cosine distances never exceed 0.7; the two are not commensurable, and a report of mean displacement that does not name its metric is uninterpretable. Only κ , being a dimensionless ratio of two quantities expressed in the same units (the upper and lower NED bounds, both themselves ratios of d_{out} to d_{DL} under the same metric), is portable across models and across metrics.

Amplification above the isometry threshold.

Under Euclidean distance at $\rho = 0.08$, the upper bound C exceeds the isometry threshold of 1 for three providers: OR_nvidia ($C = 1.115$), Azure ($C = 1.084$), and Ollama ($C = 1.029$), while Gemini (0.761), HF_e5 (0.512), and HF_bge (0.923) remain below. For the exceeding providers, at least one query-perturbation pair produces a Euclidean displacement in embedding space larger than the number of character edits, an amplification event invisible to mean-only analyses.

The threshold $C = 1$ is among the few absolute reference points in this chapter, and it divides the six models into two classes. A model with $C \leq 1$ is, in the worst case observed, an absorber: however unfavourable the query, the displacement it suffers in the latent space never exceeds the number of characters the user got wrong. A model with $C > 1$ is, at least once, an amplifier: for some query and some error it moved the vector further than the error warranted, manufacturing distance that was not present in the input. OR_nvidia, Azure and Ollama are amplifiers; Gemini, HF_e5 and HF_bge are absorbers.

This does not condemn the former, since the event is rare by construction, being a maximum over ten thousand perturbations, and Azure will go on to perform excellently at the retrieval task. It is, however, a consideration for deployment. Where a single failed query is expensive, as in a legal search, a medical lookup, or a support system without human fallback, an amplifier carries a tail risk that an absorber does not, and the three absorbers are the safer default.

Per-query and per-sample spread.

The per-query and per-sample values reported in this paragraph were saved during the experiment but are not reproduced in full here for brevity, and are not displayed in any figure. A systematic length effect is visible: at $\rho = 0.08$ under Azure cosine, per-query mean NED spans from 0.032 (“*what can you do with a degree in mathematics besides suffering?*”, 63 chars) to 0.188 (“*what is a fandango*”, 18 chars), a $6\times$ range. At the same budget the worst-case per-sample NED for Azure cosine reaches 0.587 (fandango query), while the best-case minimum is 0.006 (pesto query), a ratio of $105\times$ for the same model at the same budget.

The same structure recurs, without exception, in all six providers. Table 4.4 reports the most and least displaced query for each, and the pattern is uniform: “*what is a fandango*” is the worst-case query for every model in the study, and the best case is always one of the two longest queries. This is the prediction of Section 4.1 confirmed exactly. What varies across providers is not which query breaks but how badly: the per-query spread ranges from $4.1\times$ (OR_nvidia) to $6.1\times$ (Gemini), and the per-sample spread, which is what κ records, from $19\times$ to $294\times$.

Table 4.4. Most and least displaced query at $\rho = 0.08$ under cosine, per provider (per-query mean NED). The worst query is the same for all six models. *Spread* is the ratio between the two.

Provider	Worst query mean NED	Best query mean NED	Spread
HF_e5	0.070 <i>fandango</i> (18 ch)	0.012 <i>pesto</i> (61 ch)	$5.7\times$
Gemini	0.157 <i>fandango</i> (18 ch)	0.026 <i>bank overflow</i> (46 ch)	$6.1\times$
Azure	0.188 <i>fandango</i> (18 ch)	0.032 <i>mathematics</i> (63 ch)	$6.0\times$
Ollama	0.171 <i>fandango</i> (18 ch)	0.038 <i>stale bread</i> (64 ch)	$5.1\times$
HF_bge	0.199 <i>fandango</i> (18 ch)	0.044 <i>pesto</i> (61 ch)	$4.5\times$
OR_nvidia	0.243 <i>fandango</i> (18 ch)	0.059 <i>pesto</i> (61 ch)	$4.1\times$

The $105\times$ figure for Azure warrants being spelled out, since it concerns the model a practitioner is most likely to select by default. At three errors on a forty-character query, a user asking “*why should a pesto recipe with walnuts be considered a crime?*” and mistyping it will have the query displaced almost imperceptibly, and the correct documents will be retrieved. A user asking “*what is a fandango*” and mistyping it will have the query displaced a hundred times further, in relative terms, for an error of the same magnitude. The same model, on the same day, with the same degree of carelessness, produces two entirely different experiences.

The full bi-Lipschitz summary at the reference budget $\rho = 0.08$, which lies at the centre of the most diagnostically informative regime $\rho \in [0.05, 0.15]$ and is used consistently throughout the rest of this chapter, is given in Table 4.5. In the table the providers are ordered by their mean κ across the four distances, from the most uniform (Gemini) to the least (HF_bge). Because κ is a ratio of the same metric across query-perturbation pairs it is dimensionless, so the same scale applies to every provider and distance regardless of embedding dimension: $\kappa = 1$ would denote a perfect isometry,

and larger values an increasingly uneven response. Each cell is shaded by its distortion band on this multiplicative scale, from green for the lowest distortion to red for the most severe, so that the regions of metric collapse stand out at a glance.

Distortion Index and Robustness Ranking

As anticipated in Section 4.2, mean NED summarises the average response of each model, but it conceals query-level variability: two models with identical mean NED may differ dramatically in whether that displacement is spread uniformly across queries or concentrated on a few adversarial ones. The distortion index $\kappa(\rho) := C(\rho)/c(\rho)$ captures this: it is large when some query-perturbation pairs produce higher displacement ratios than others, regardless of the mean.

Stated in one sentence, κ is the ratio between the most and the least unfavourably treated query. A κ of 10 means that the worst-treated query suffers ten times more than the best-treated one for an error of the same magnitude; a κ of 300 means that the model is, for practical purposes, a lottery. Only $\kappa = 1$ would be perfectly uniform, and no model approaches it.

Distortion across metrics and across providers.

The key observation from Table 4.5 is that κ varies far more across metrics within the same provider than across providers within the same metric. For every provider, cosine κ is three to five times higher than Euclidean κ : at $\rho = 0.08$, HF_bge has $\kappa_{\text{cos}} = 293.7$ but $\kappa_{\text{euc}} = 29.7$. This is not a sign that cosine retrieval is unreliable (as shown in Section 4.3, it is in fact the most stable metric for retrieval). Rather, it reflects a structural difference: cosine measures angular displacement, which is highly sensitive to which specific word is corrupted and therefore query-dependent, whereas ℓ_p norms aggregate all coordinate deviations and are more uniform. The additional spike at low ρ (where κ peaks for all providers) arises because at $\rho = 0.01$ every query receives exactly one edit: a single mutation on a semantically critical short query versus a peripheral character in a long query produces the maximum possible NED variance.

This is the most counter-intuitive result in the chapter, and a reader taking the κ column at face value would draw precisely the wrong conclusion from it. Under cosine the models appear poor, with distortions in the hundreds; under Euclidean the same models appear respectable, with distortions in the tens. Yet Section 4.3 will show cosine to be the metric under which retrieval in fact works.

The resolution is that κ and retrieval measure different things. κ is sensitive to which word has been corrupted: cosine registers, correctly, that corrupting *fandango* in a three-word query is a semantic catastrophe while corrupting one letter of *mathematics* in a ten-word question is not, and it reports a large spread between the two. That spread is not unreliability but discrimination: the metric is performing its function of distinguishing a harmful error from a harmless one. The ℓ_p norms report a smaller spread not because they are more stable but because they are less discriminating, averaging a semantic catastrophe and a harmless slip into comparable figures. A high κ is thus the signature of a metric

sensitive to meaning, not of a metric that is broken, and should be read as a diagnostic of which queries are at risk rather than as a verdict on the metric itself.

Table 4.5. Empirical bi-Lipschitz summary at $\rho = 0.08$. $C(\rho)$: upper Lipschitz constant (U-NED); $c(\rho)$: lower constant (L-NED); $\kappa = C/c$: distortion index. Cell background:

$\kappa < 15$ 15–25 25–50 50–100 $\kappa \geq 100$.

Embedding	Metric	ρ	$C(\rho)$	$c(\rho)$	$\kappa(\rho)$
Google gemini-001	<i>Chebyshev</i>	0.08	0.0876	0.0075	11.8
	<i>cosine</i>	0.08	0.2899	0.0150	19.3
	<i>Euclidean</i>	0.08	0.7614	0.0783	9.7
	<i>Manhattan</i>	0.08	33.223	3.2636	10.2
HF intfloat-e5-large	<i>Chebyshev</i>	0.08	0.0640	0.0036	18.0
	<i>cosine</i>	0.08	0.1309	0.0028	47.3
	<i>Euclidean</i>	0.08	0.5117	0.0333	15.4
	<i>Manhattan</i>	0.08	13.145	0.8512	15.4
OR nvidia-nemotron	<i>Chebyshev</i>	0.08	0.0944	0.0051	18.4
	<i>cosine</i>	0.08	0.6210	0.0115	54.0
	<i>Euclidean</i>	0.08	1.1146	0.0709	15.7
	<i>Manhattan</i>	0.08	40.160	2.5542	15.7
Azure	<i>Chebyshev</i>	0.08	0.0879	0.0036	24.6
	<i>cosine</i>	0.08	0.5872	0.0056	105.0
	<i>Euclidean</i>	0.08	1.0836	0.0473	22.9
	<i>Manhattan</i>	0.08	46.695	2.0533	22.7
Ollama nomic-embed	<i>Chebyshev</i>	0.08	0.1417	0.0046	30.7
	<i>cosine</i>	0.08	0.5298	0.0031	169.0
	<i>Euclidean</i>	0.08	1.0294	0.0354	29.1
	<i>Manhattan</i>	0.08	22.674	0.7874	28.8
HF BAAI-bge-m3	<i>Chebyshev</i>	0.08	0.1154	0.0033	34.5
	<i>cosine</i>	0.08	0.4264	0.0015	293.7
	<i>Euclidean</i>	0.08	0.9234	0.0311	29.7
	<i>Manhattan</i>	0.08	23.271	0.7921	29.4

Robustness ranking.

Figure 4.12 ranks providers by mean per-query κ at $\rho = 0.08$. Since κ measures the ratio between the most and least displaced query-perturbation pair, lower values (approaching the ideal $\kappa = 1$) indicate more uniform and robust behaviour, while higher values reveal greater sensitivity disparities across queries. Providers with the highest bars are therefore

the least robust.

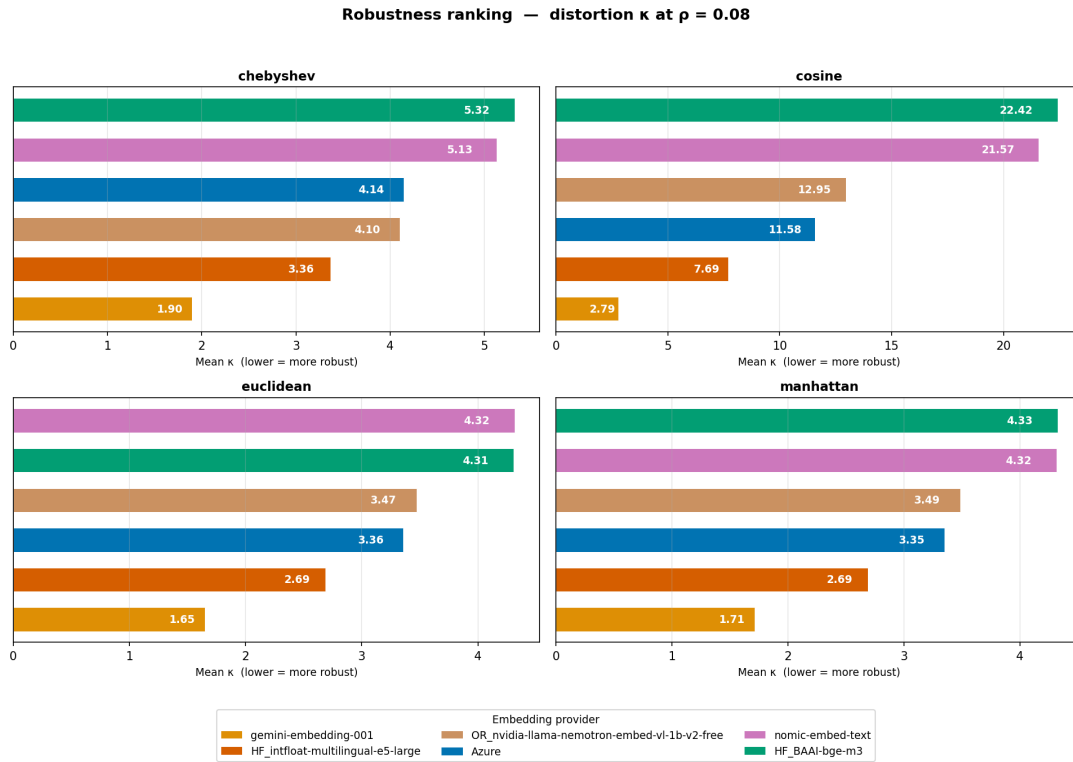


Figure 4.12. Providers robustness ranking at $\rho = 0.08$ by distance function; bars show mean per-query distortion $\bar{\kappa}(\rho)$; lower values indicate more uniform response across queries.

Gemini has the lowest distortion across all four distances ($\bar{\kappa}_{\text{cos}} = 2.79$, $\bar{\kappa}_{\text{euc}} = 1.65$), confirming that its embedding space responds uniformly regardless of which query is perturbed. HF_e5 follows ($\bar{\kappa}_{\text{cos}} = 7.69$, $\bar{\kappa}_{\text{euc}} = 2.69$), roughly $1.5\times$ lower than Azure (11.58) and OR_nvidia (12.95). Ollama (21.57) and HF_bge (22.42) are at the bottom.

Uniform and selective failure modes.

Two tensions in the data deserve emphasis before the ranking is interpreted. The low κ of Gemini coexists with a relatively high mean NED at low budgets (0.125 at $\rho = 0.01$): it displaces significantly on average, but uniformly across queries. Azure is the mirror image, with a low mean cosine NED (0.077 at $\rho = 0.08$) but a global $\kappa_{\text{cos}} = 105$, so that its average stability conceals a long tail of adversarially sensitive pairs. HF_e5 alone combines a low mean NED with comparatively low distortion under both cosine and Euclidean.

The ranking therefore sorts the six models not into successes and failures but into two philosophies of failure, and the choice between them is a genuine engineering decision rather than a matter of one being superior.

Gemini and HF_e5 fail uniformly. They displace all queries by a comparable amount, the worst query of Gemini suffering less than three times more than its best, so that as input quality deteriorates everything degrades a little together and nothing collapses

abruptly. This is the profile to select where the cost of a failure is roughly constant across queries and where predictability outweighs peak performance, as in a public-facing search interface in which any query might prove to be the important one.

Azure fails selectively, its worst query suffering a hundred times more than its best. The failure mode of a selective model is the rare catastrophe: the average user notices nothing, while a small minority find that the system has ceased to function for them entirely. This is acceptable, and may even be optimal, where the queries the system will receive are known in advance and can be tested, as in a closed-domain documentation assistant, and it is precisely why Azure will dominate the in-domain RAG results of Section 4.3 despite its distortion figure. It is a poor choice where the query distribution is open and unmonitored.

OR_nvidia occupies an awkward middle position. Its distortion is moderate ($\bar{\kappa}_{\cos} = 12.95$) but its mean NED is the worst of the six, so that it is neither uniform nor accurate: it displaces every query substantially, and somewhat unevenly. It is defensible only where its clean-query retrieval quality, which the present study does not measure, justifies the cost, and in any case behind a spell-checker.

Ollama and HF_bge fail selectively without the compensating strengths, combining the worst distortion in the set with unexceptional mean displacement. On the evidence of this chapter there is no regime in which they are the appropriate choice for noisy input, a conclusion that Section 4.3 will only sharpen.

Geometric Saturation Analysis (t-SNE)

The t-SNE projections provide a qualitative view of the saturation trajectory, showing perturbed embeddings (coloured by ρ), the clean query (green star), and per-budget centroid chains. Since t-SNE does not preserve metric distances, interpretation focuses on centroid trajectories and local density patterns rather than absolute distances. The projections are computed per provider directly in embedding space, so they reflect its intrinsic geometry; their relation to distortion regimes is captured by the cosine distortion κ , which determines the ordering in Figure 4.13.

Each panel is a flattened map of one model’s internal space for a single query. The green star marks the query as the user intended to type it; the coloured dots are the five hundred ways in which it might have been mistyped, dark for light corruption and yellow for heavy; the arrows trace where the average corrupted query lies as the typing deteriorates. The question to be put to each panel is whether the average drifts away smoothly or shifts discontinuously.

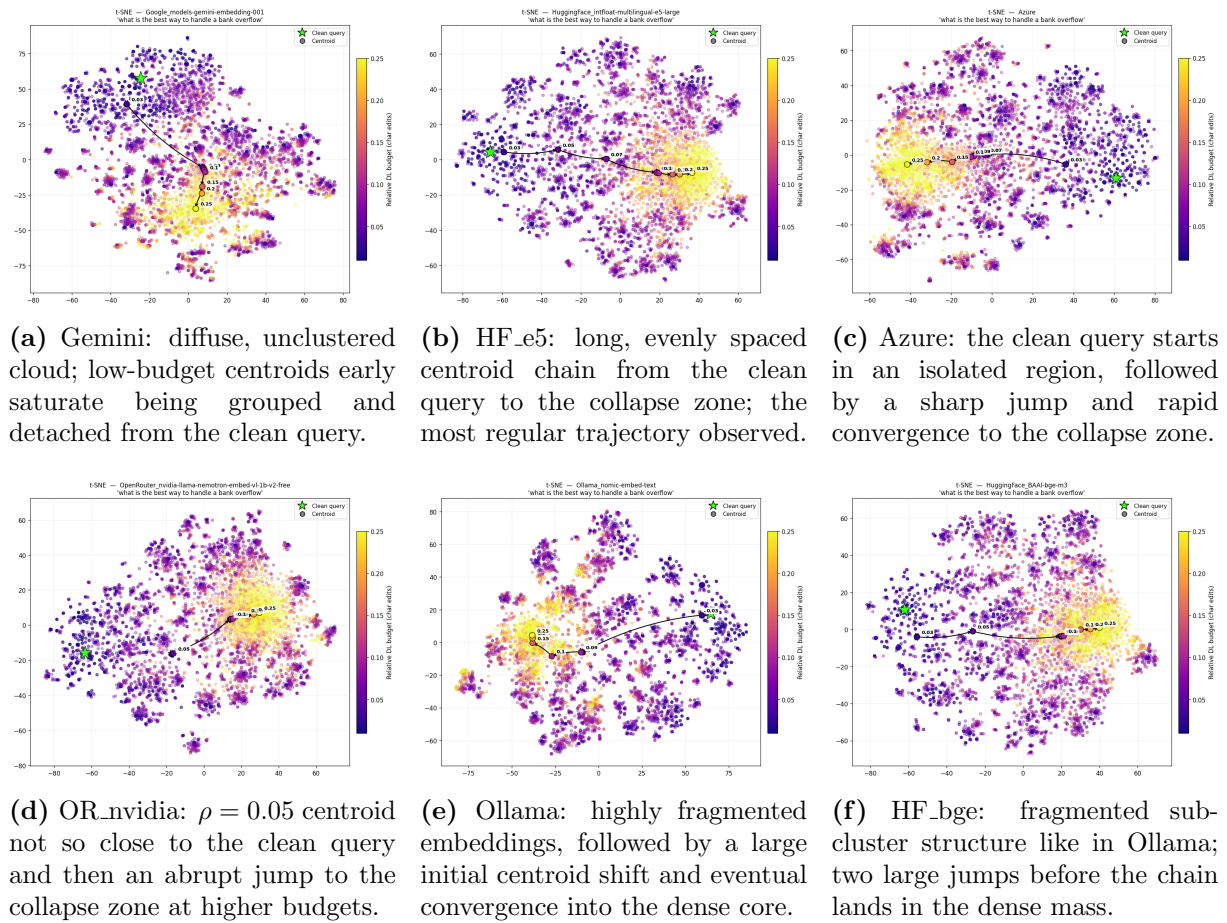


Figure 4.13. t-SNE projections of the query “what is the best way to handle a bank overflow” across six providers, isolating model-dependent effects. Colours indicate perturbation budget ρ , the green star marks the clean query, and arrows connect consecutive centroids. Panels are ordered by increasing cosine distortion.

Across all six panels the clean query embedding sits in a sparse, low-density region while the dense yellow mass of high-budget perturbations concentrates elsewhere, and the centroid chain points from the former toward the latter. This collapse zone is a compact attractor to which high-budget perturbed queries converge regardless of original content, and it is the geometric correlate of the saturation measured in Section 4.2: the colour gradient along each chain (dark purple near the clean query, yellow at the attractor) confirms that the convergence is monotone in ρ .

The existence of this collapse zone accounts for the particular manner in which badly corrupted queries fail. Every model possesses a region of its internal space into which heavily corrupted queries are drawn irrespective of what they originally said: a ruined question about Rulex filters and a ruined question about fandangos converge to roughly the same place. At that point the model has not misunderstood the query; it has ceased to represent it as a query at all, and represents it as noise. Retrieval from that region is essentially arbitrary, which is why no amount of downstream processing recovers a query that has reached it. What distinguishes the models is not whether they possess such a

zone but how rapidly a query descends into it.

The trajectories fall into three regimes that correlate with the cosine distortion κ of each provider, distinguished by how *evenly* the centroid chain advances rather than by its overall direction, which always points toward the collapse zone.

- **First regime: smooth, evenly spaced progression.** The displacement accumulates gradually as edits distribute across tokens. The clearest case is HF_e5 (Figure 4.13b), whose chain advances in roughly equal steps from the clean query to the collapse zone.

This is the well-behaved case, and the one for which a system can be designed: query quality deteriorates in proportion to typing quality, so that a lightly corrupted query still retrieves largely correct documents and the degradation curve is known and gradual.

- **Second regime: a single early stride, then a tighter progression.** The chain takes one large step out of the low-density corner and then advances in small increments. This is the Azure pattern (Figure 4.13c): the centroid moves little up to $\rho = 0.03$, makes a single large jump into the populated centre, and from there progresses smoothly and regularly into the collapse zone, so Azure is irregular only on that first stride rather than throughout. The jump here is not caused by a fragmented background but by the isolated, low-density corner in which Azure places this ambiguous query: once the perturbation is large enough to leave that corner, the embedding lands in the populated centre and moves steadily thereafter. OR_nvidia (Figure 4.13d) is a sharper version of the same shape, with the $\rho = 0.05$ centroid still beside the clean query and the next recorded centroid already in the dense mass.

This is the signature of a model that assigns distinctive queries their own isolated region. It constitutes a strength so long as the query survives intact, since a well-isolated query retrieves unambiguously, and it is precisely why Azure leads the in-domain retrieval results of Section 4.3. The price is that once an error displaces the query from that region it lands among the crowd, with no intermediate stage. Azure and OR_nvidia are therefore models to deploy where queries are distinctive and largely clean, rather than terse and noisy.

- **Third regime: large discontinuous jumps separated by near-stationary stretches.** This is the signature of providers whose background is fragmented into dense sub-clusters with visible gaps between them. The mechanism is straightforward: perturbations too small to push the query out of its current sub-cluster barely move the centroid, while one that crosses a gap moves it abruptly by the full width of that gap. This differs from the Azure case, where the single large stride stems from an isolated starting position rather than from a fragmented background. Ollama (Figure 4.13e) shows one such jump, HF_bge (Figure 4.13f) two, and both are the geometric counterpart of the large angular displacements behind their high cosine κ .

This is the hazardous case. The latent space is broken into islands, and an error that displaces a query from its island deposits it in an unrelated neighbourhood with no intermediate stage. There is no graceful degradation against which to tune: the query is either intact or it is elsewhere entirely. This is the geometric picture underlying the catastrophic Ollama retrieval collapse of Section 4.3, in which a single additional character replaces the entire retrieved document set.

Summary of the three regimes.

Gemini (Figure 4.13a) and HF_e5 produce diffuse, unclustered backgrounds in which centroids move smoothly, the geometric expression of their low cosine distortion ($\bar{\kappa} = 2.79$ and 7.69). At the opposite end, Ollama and HF_bge fragment into dense sub-clusters separated by visible gaps, the topology that lets small perturbations trigger the large angular jumps behind their high distortion ($\kappa = 169$ and 294). Azure sits between: a moderately structured background, but with this ambiguous query placed in an isolated corner from which even a small perturbation displaces it toward the populated centre.

The contrast may be put in a single sentence: a smooth embedding space degrades, whereas a fragmented one breaks. The two are not points on the same scale, and a model that fails by breaking cannot be rescued by adjusting the tolerable noise level, since there is no noise level at which it fails gently.

4.3 RAG Quality Results

This section evaluates how the embedding instability characterised in Section 4.2 propagates to the downstream RAG pipeline. The pipeline architecture, evaluation protocol, query set, and metrics are described in Section 3.4; this section reports results only. Two end-to-end quantities are measured: retrieval stability (Jaccard overlap between the chunk-ID sets retrieved for the clean and perturbed query) and answer quality (BERTScore F1 between the perturbed-query and clean-query answers).

The two ask different questions. Jaccard asks whether the system is still consulting the same documents: a value of 1 indicates that the corrupted query retrieved exactly the same passages as the clean one, and 0 that it retrieved none of them. BERTScore asks whether the user received the same answer. The interest of this section lies in the fact that the two do not move together.

The evaluation uses the four probe queries of Table 3.4, partitioned into an in-domain pair targeting the Rulex documentation directly and an out-of-domain pair whose semantics fall outside the knowledge base. The six complete embedding providers of Section 4.2 are evaluated across all four distance functions and the full 13-point ρ grid, with $N_{\text{typo}} := 30$ perturbed variants per cell. A single LLM backend (Azure `gpt-4o-mini`) generates all answers, isolating the effect of retrieval degradation from generator variability. The in-domain and out-of-domain partitions are reported separately throughout, because the two regimes behave qualitatively differently.

The two partitions correspond to two situations a deployed assistant encounters daily. In-domain queries are those the system was built to answer: “*how do i filter my data in rulex factory?*” has a correct answer in the documentation, and the correct behaviour is to find it. Out-of-domain queries are those it was not: no Rulex manual page discusses fandangos, and the correct behaviour is to say so. Both must survive typographical noise, and they fail in instructively different ways.

Aggregate Retrieval and Answer Degradation

The aggregate picture, averaged over all providers, distance metrics, and queries within each domain partition, is shown in Figure 4.14. Two observations dominate.

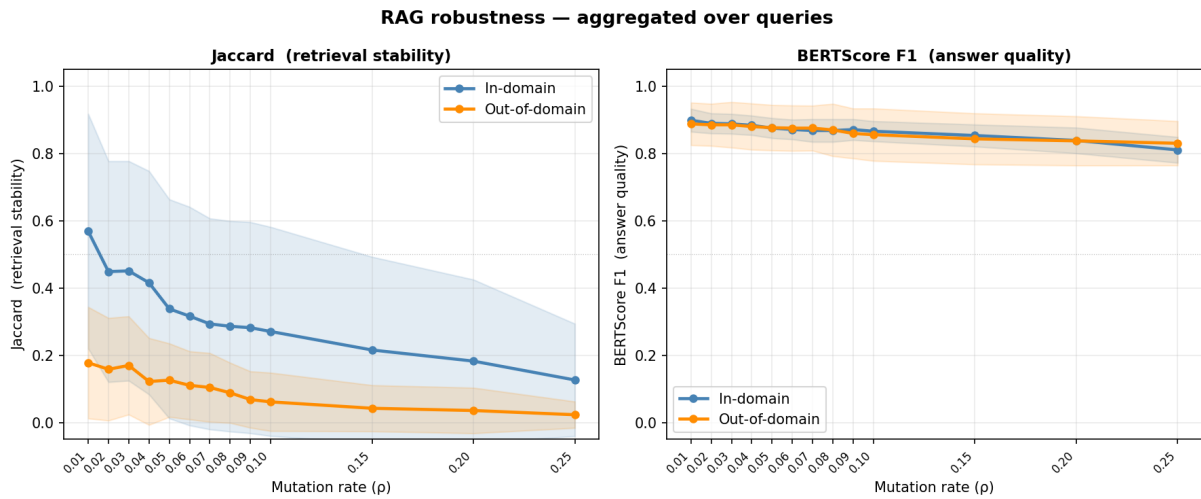


Figure 4.14. Aggregate RAG robustness as a function of mutation rate ρ , averaged over the five complete providers, the four distance metrics, and the two queries in each domain partition. Left: Jaccard retrieval overlap with the clean query. Right: BERTScore F1 of the perturbed answer against the clean answer. Shaded bands denote ± 1 standard deviation across queries within each partition. Blue: in-domain; orange: out-of-domain.

Divergent degradation of retrieval and answer quality.

Aggregate in-domain Jaccard falls from 0.59 at $\rho = 0.01$ to 0.29 at $\rho = 0.08$ and 0.12 at $\rho = 0.25$, a relative collapse of roughly 80% over the tested range. Over the same range, in-domain BERTScore declines only from 0.90 to 0.87 to 0.81, a relative drop of about 10%. The retriever therefore loses the large majority of its original document set well before the generated answer shows any comparable degradation. This dissociation is the central finding of the RAG module: the answer the user receives is far more robust to typographical noise than the retrieval step that ostensibly produces it. The mechanism is that the generator, conditioned on a partially overlapping but still topically-relevant set of chunks, reconstructs an answer that remains semantically close to the clean-query answer even when up to half of the specific retrieved chunks have changed. A plausible contributing factor is the redundancy structure of the corpus: the Rulex knowledge base

is indexed in both HTML and plain-text format, so the same content is available in two collections. A perturbed query may shift to a different set of chunk IDs while still retrieving semantically equivalent passages from the parallel collection, allowing the generator to produce an answer of comparable quality. This is consistent with the out-of-domain behaviour, where no redundant coverage exists and the BERTScore curves do not diverge from the in-domain ones (the generator converges on the same “not found” response in both cases).

The finding is at once favourable and cautionary, and it is worth being clear as to which is which. For the user it is favourable: the pipeline proves considerably more robust than its weakest component. At four typographical errors the retriever has already discarded seventy per cent of the documents it ought to have found, and the user nonetheless receives an answer ninety per cent as good as the correct one, the language model reconstructing something serviceable from the fragments that survive.

For the engineer the same fact is a caution: retrieval quality is very nearly invisible from outside the system. A team monitoring end-user answer quality alone would observe a ten per cent decline and conclude that matters were broadly acceptable, while the retrieval layer beneath had quietly lost most of its accuracy. Two consequences follow. Retrieval must be monitored directly, with a metric such as Jaccard, and never inferred from answer quality. And the robustness observed here is borrowed rather than earned: it depends upon the generator having something usable in its context window, and upon the corpus containing the same information in more than one place. In a sparser knowledge base, where each fact appears exactly once, the retrieval collapse documented here would translate directly into incorrect answers.

The gap between domain partitions.

In-domain Jaccard starts more than $2.5\times$ higher than out-of-domain at the lowest budget (0.59 vs 0.20 at $\rho = 0.01$), and the two curves never cross. The out-of-domain queries begin with a weak, poorly-anchored retrieval match (no chunk in the knowledge base is genuinely relevant to a question about bank overflows or fandangos) and a single character edit is enough to scatter the already unstable retrieval set. The wide in-domain variance band, spanning nearly the full $[0, 1]$ interval at low ρ , reflects the strong query-to-query and provider-to-provider heterogeneity that the aggregate mean conceals, and which the stratified views below disentangle. Notably, the BERTScore panels for the two partitions are nearly indistinguishable across the entire range, confirming that answer quality is essentially decoupled from whether the query was in-domain: the generator produces a stable, self-consistent answer regardless of retrieval relevance.

The out-of-domain result is best read as a null result requiring verification rather than as a finding in its own right. Where a query has no answer in the knowledge base, the retriever is already selecting whatever happens to lie nearest, and a typographical error merely alters which irrelevant documents it selects; there is no stable retrieval to destabilise. What matters is the behaviour downstream, and it is reassuring for deployment: the answer remains stable regardless, since *“I could not find this in the documentation”* is a stable answer whichever irrelevant passages were consulted in reaching it. Typographical noise does not, in other words, induce an assistant to begin

fabricating answers to questions it cannot answer; it continues to decline them.

Distance-Metric Dependence

Stratifying the aggregate by distance function, while still averaging over providers and queries, reveals that the retrieval distance function is the single largest controllable factor in RAG robustness. Figure 4.15 reports this view for both domain partitions.

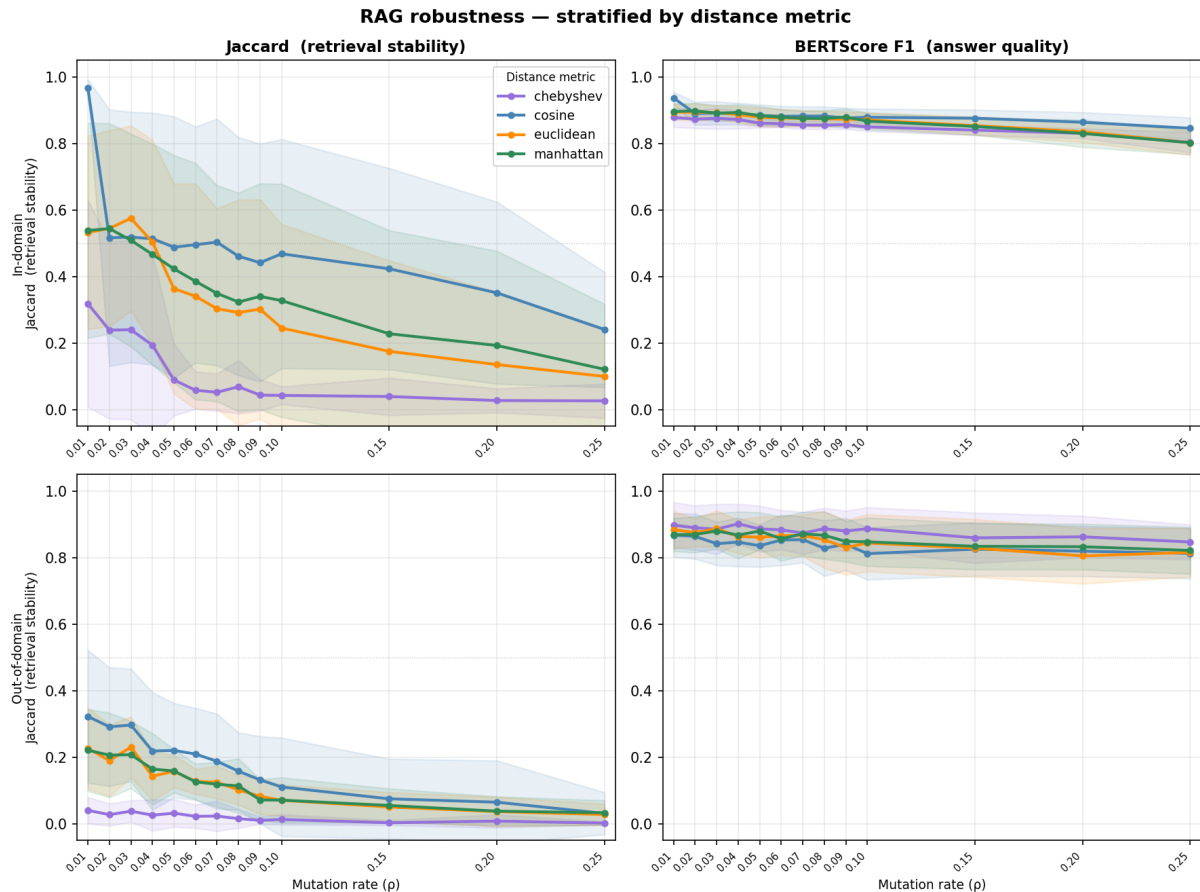


Figure 4.15. RAG robustness stratified by distance function, averaged over the six complete providers and the two queries in each partition. Top row: in-domain; bottom row: out-of-domain. Left column: Jaccard; right column: BERTScore F1. Shaded bands denote ± 1 standard deviation across embedding models. Colour encodes the distance function (blue: cosine; orange: euclidean; green: manhattan; purple: chebyshev).

Cosine as the most stable retrieval metric.

Under cosine retrieval, in-domain Jaccard remains the highest of all four distances across the entire ρ range, starting at 0.97 at $\rho = 0.01$ and remaining at or above 0.46 through $\rho = 0.08$ before declining to 0.24 at $\rho = 0.25$. The three ℓ_p metrics degrade markedly faster. Euclidean and Manhattan, which produce nearly identical curves throughout, start barely above 0.53 at the lowest budget and fall below 0.15 by $\rho = 0.20$. Chebyshev is

by far the least stable metric for retrieval: its in-domain Jaccard starts at only 0.32 at $\rho = 0.01$ and collapses below 0.07 by $\rho = 0.08$.

Of all the results reported in this chapter, this one carries the highest ratio of benefit to cost, since the retrieval distance is a single configuration parameter in any vector database. Stated concretely: under cosine a corrupted query still recovers about half the correct documents at four typographical errors, whereas under Chebyshev it recovers almost none, having already lost 93% of the retrieval set at the budget where cosine retains half of it, and retrieving only a third of the correct documents even at a single error.

The mechanism is the one anticipated in Section 4.2. ℓ_∞ inspects only the single most disturbed coordinate of the embedding, so that one dimension going wrong suffices to reshuffle the entire neighbour list, whereas cosine, which distributes the disturbance across the whole direction of the vector, is far harder to displace. Euclidean and Manhattan fall predictably between, losing roughly twice as much as cosine at every budget. The recommendation is accordingly unambiguous and independent of the embedding model chosen: retrieval should be performed under cosine distance. A system currently retrieving under Euclidean may be switched at no cost and will roughly double its tolerance to typographical noise.

Reconciliation with the distortion analysis.

This ordering is consistent with the embedding-stability results of Section 4.2, but requires careful interpretation. Although cosine exhibited the *highest* distortion index κ in the geometric analysis, that distortion measures the spread between the most and least adversarial query-perturbation pairs, not the rank stability of the top- k retrieval. For the retrieval task what matters is whether the relative ordering of candidate chunks is preserved, and the angular metric preserves that ordering far better than norm-based metrics, which are dominated by the magnitude excursions that the bi-Lipschitz upper bound $C(\rho)$ already showed to exceed isometry for several providers. The Chebyshev collapse is the retrieval-side counterpart of the ℓ_∞ behaviour noted in Section 4.2: a single large coordinate deviation amplified by ℓ_∞ is enough to reorder the nearest neighbours and replace the retrieved set entirely.

The apparent paradox, cosine being at once the most distorted and the best performing, dissolves once one observes that retrieval is indifferent to how far a query has moved and sensitive only to whether it has moved past another candidate. A document is either retrieved or it is not; what matters is the ordering of the candidates, not the absolute distances between them. A metric may shift every distance by a large factor and leave the ordering untouched, which is broadly what cosine does, or shift distances only slightly and still exchange the top of the list, which is what ℓ_∞ does when a single coordinate misbehaves. The general lesson, worth carrying beyond the present study, is that geometric stability in the embedding space is a proxy for retrieval stability, and an imperfect one: both must be measured.

Metric independence of answer quality.

The BERTScore panels show almost no metric dependence in either partition: all four

curves cluster within a few hundredths of each other and decline only gently. The choice of distance function thus strongly governs which chunks are retrieved but has little bearing on the quality of the answer ultimately generated, reinforcing the retrieval/generation dissociation established in Section 4.3.

Here again the damage is invisible to the user. A team that had misconfigured its vector store to retrieve under ℓ_∞ , and was consequently recovering almost none of the correct documents under mildly noisy input, would observe answer quality essentially indistinguishable from that of a correctly configured system. The misconfiguration would leave no trace in any user-facing metric, which constitutes a strong argument for evaluating retrieval on its own terms rather than trusting end-to-end answer scores to surface upstream failures.

Provider Dependence

The complementary stratification, by embedding provider with curves for each distance function, isolates the role of the embedding model. The in-domain partition is reported in Figures 4.16 and 4.17, and the out-of-domain partition in Figures 4.18 and 4.19.

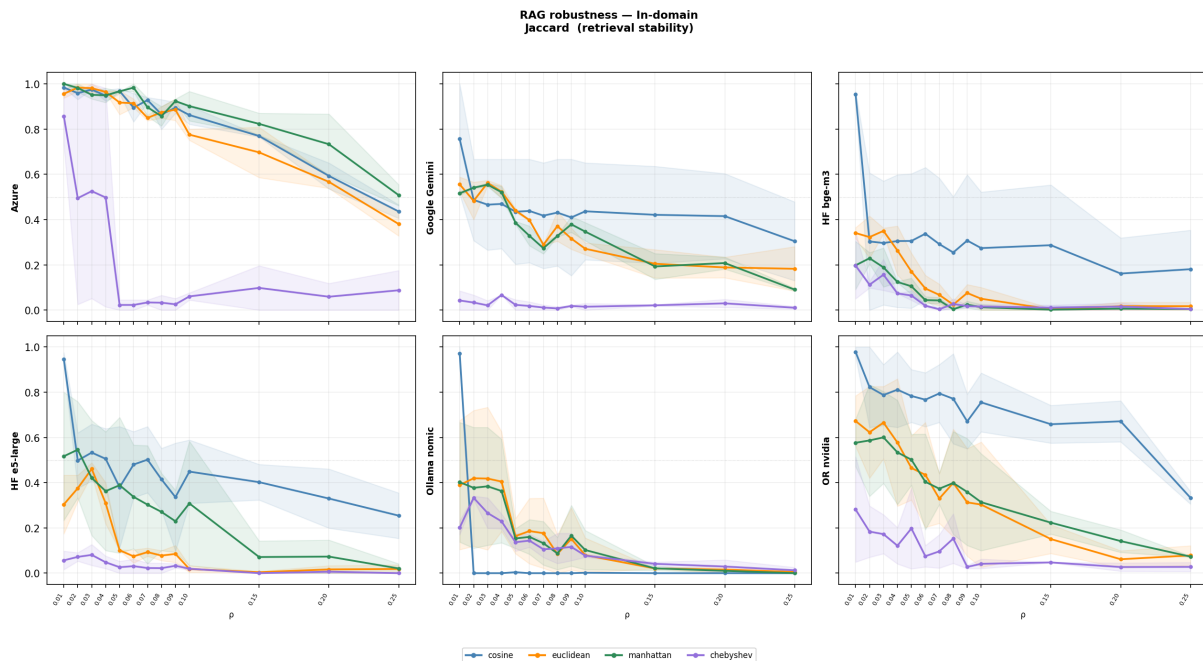


Figure 4.16. In-domain retrieval stability (Jaccard) by provider. Curves: four distance metrics; bands: ± 1 std across the two in-domain queries.

In-domain retrieval stability across providers.

The provider stratification reveals a spread in retrieval stability that the aggregate conceals, governed almost exclusively by the cosine curve. Restricting attention to cosine in-domain Jaccard, Azure is the most stable provider (0.87 at $\rho = 0.08$, 0.44 at

$\rho = 0.25$), followed by OR_nvidia (0.77, 0.33), HF_e5 (0.42, 0.25) and HF_bge (0.26, 0.18). The most striking behaviour is that of Ollama nomic-embed-text, whose cosine Jaccard collapses from 0.97 at $\rho = 0.01$ to 0.00 at $\rho = 0.02$ and remains there thereafter. Rather than a gradual degradation, a single character edit beyond the smallest budget replaces the entire retrieved set. This reflects the pathological cosine geometry identified in Section 4.2 (global $\kappa_{\cos} = 169$) and the fragmented sub-cluster topology observed in Section 4.2: a perturbed query quickly crosses a cluster boundary and retrieves a completely different neighbourhood.

The magnitude of the Ollama collapse is worth stating without formalism. Presented with “*how do i filter my data in rulex factory?*”, the model retrieves the correct passages; presented with “*how do i filter my data in rulex factory?*”, which differs by a single character, it retrieves none of them. Not fewer, but none. The Jaccard overlap falls from 0.97 to exactly 0.00, and remains there at every higher budget.

There is no intermediate regime here in which the system is partially degraded and might be patched or tuned; there is a discontinuity, and it occurs at the second error. The geometric explanation was given in Section 4.2: the latent space of Ollama is fragmented into islands, and one additional character suffices to displace a query from its island onto an unrelated one. The operational conclusion is correspondingly sharp: the model is unsuited to deployment before a human typist, and remains serviceable only where queries arrive validated or are generated programmatically. HF_bge, which exhibits the same pathology, degrades less abruptly but recovers only 0.26 of the correct documents at four errors, and carries the same warning in attenuated form.

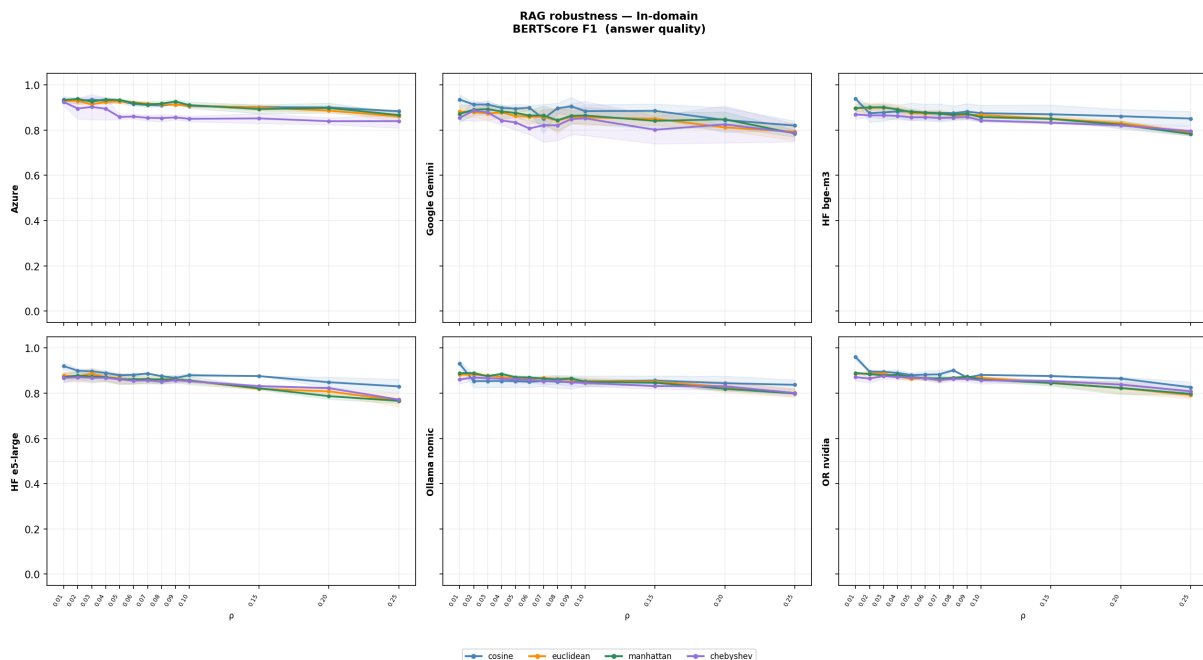


Figure 4.17. In-domain answer quality (BERTScore F1) by provider. Curves and bands as in Figure 4.16.

Divergence between embedding stability and retrieval stability.

The relative ranking of providers under cosine retrieval, with Azure and OR_nvidia clearly ahead of the two HuggingFace models and Ollama at the floor, only partially mirrors the embedding-stability ranking of Table 4.6, where HF_e5 and Gemini were judged the best overall and Azure merely good. The discrepancy is informative: the embedding-level ranking rewarded geometric uniformity (low distortion across the full query set), whereas the retrieval task rewards preservation of the top- k neighbour ordering for the specific in-domain queries. Azure, despite its high global distortion, evidently places the in-domain Rulx queries in well-separated, stable neighbourhoods whose ordering survives moderate perturbation. This confirms that embedding stability and retrieval stability are related but non-equivalent criteria, and that a provider should be selected against the task that will actually be deployed. The BERTScore panels (Figure 4.17) remain high and almost flat for every provider, confirming that the wide spread in retrieval does not carry over to the generated answer.

Section 4.2 predicted that Azure would behave selectively: excellent on most queries, catastrophic on a few, and safe only where the query set is known in advance. The RAG results vindicate that prediction in the manner most favourable to Azure. In the closed-domain setting, where the query set is known, Azure proves the best retriever of the six by a wide margin, still recovering 87% of the correct documents at four errors, twice what HF_e5 manages, while its ostensibly alarming distortion never manifests, since none of the deployed queries fall among its adversarial cases. The isolated low-density corner visible in its t-SNE panel (Figure 4.13c) proves to be an asset here: a query that sits alone retrieves unambiguously.

The nuance generalises well beyond the present study. A poor global distortion score does not entail that a model will perform badly; it entails that the model has some queries on which it performs badly. Whether this matters depends entirely upon whether those queries are the ones the system will receive. In a closed domain they can be identified and tested; in an open one they cannot, and the risk is real. The warning to the practitioner is therefore not to avoid high-distortion models but to evaluate them against one's own queries, since the aggregate benchmark will not reveal whether a particular query set falls within a model's blind spot. The mirror-image warning applies to HF_e5 and Gemini, which appear strongest on the embedding metrics yet trail Azure by a factor of two on the retrieval task actually deployed: geometric uniformity is no guarantee of retrieval quality either.

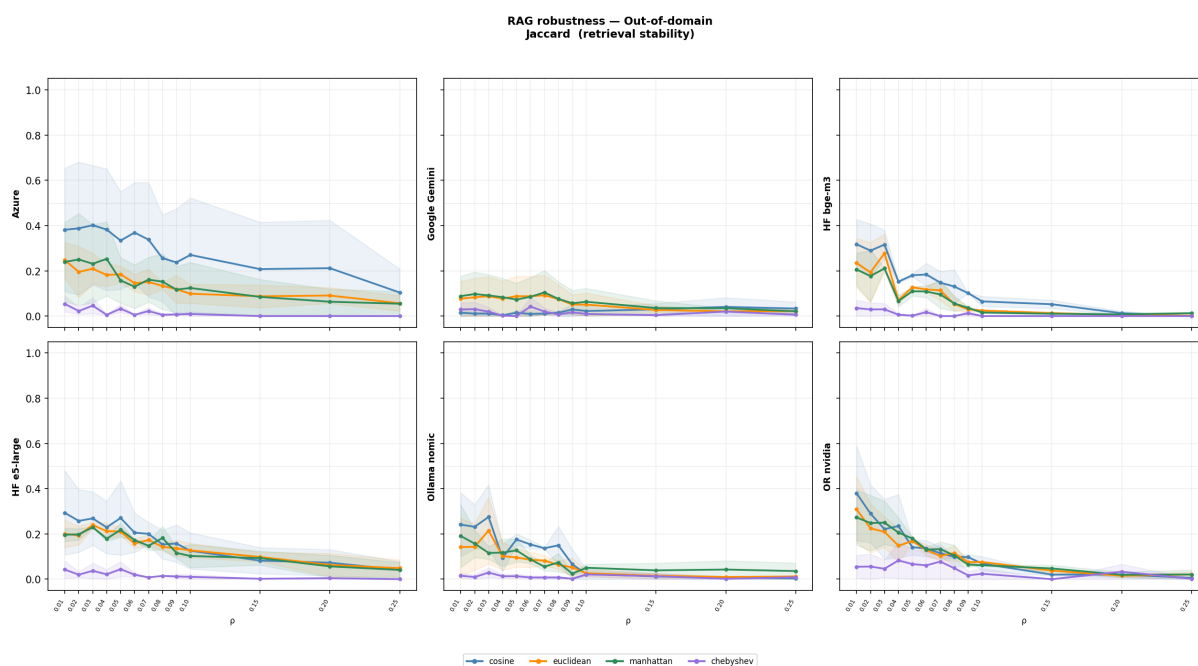


Figure 4.18. Out-of-domain retrieval stability (Jaccard) by provider. Curves and bands as in Figure 4.16.

Out-of-domain retrieval.

In the out-of-domain partition the provider differences in retrieval largely vanish: all six models start with low Jaccard (≤ 0.4 even under cosine) and converge toward zero by $\rho = 0.10$ – 0.15 under all metrics, with Azure cosine retaining a marginally higher but still weak overlap. This is expected, since none of the providers can stably retrieve documents for queries with no genuine match in the knowledge base.

The choice of provider is therefore immaterial for queries the knowledge base cannot answer. No degree of embedding quality compensates for the absence of a correct document, and the appropriate design response is not a better embedding model but an out-of-domain detector: a component that recognises the query as unanswerable before an LLM call is expended upon it.

Out-of-domain answer quality.

The BERTScore panels (Figure 4.19) remain high and flat for every provider, with average out-of-domain answer similarity ranging from 0.81 (OR_nvidia) to 0.91 (Ollama). The fact that Ollama achieves the *highest* answer similarity while suffering the *worst* retrieval collapse is the sharpest single illustration of the retrieval/generation dissociation: even when the retrieved set is completely replaced, the generator produces an answer that stays close to the clean-query answer, because for an unanswerable out-of-domain query the model converges on a similar “cannot find relevant information” response regardless of which irrelevant chunks were retrieved.

Taken at face value, the figure of 0.91 would make Ollama the best of the six models;

it is in fact the worst in the study. The explanation is straightforward. The retrieval of Ollama has collapsed so completely that it returns irrelevant material; the language model, presented with irrelevant material, correctly reports that it can find nothing; and since this is also what it reports when presented with the correct-but-irrelevant documents retrieved from the clean query, the two answers coincide almost exactly. Ollama scores highly, in short, because it fails consistently.

The methodological warning could not be sharper. An end-to-end answer-similarity score is capable of rewarding a system for failing reliably, and any evaluation of a RAG pipeline that reports answer quality without inspecting what was retrieved may crown the weakest component in the stack.

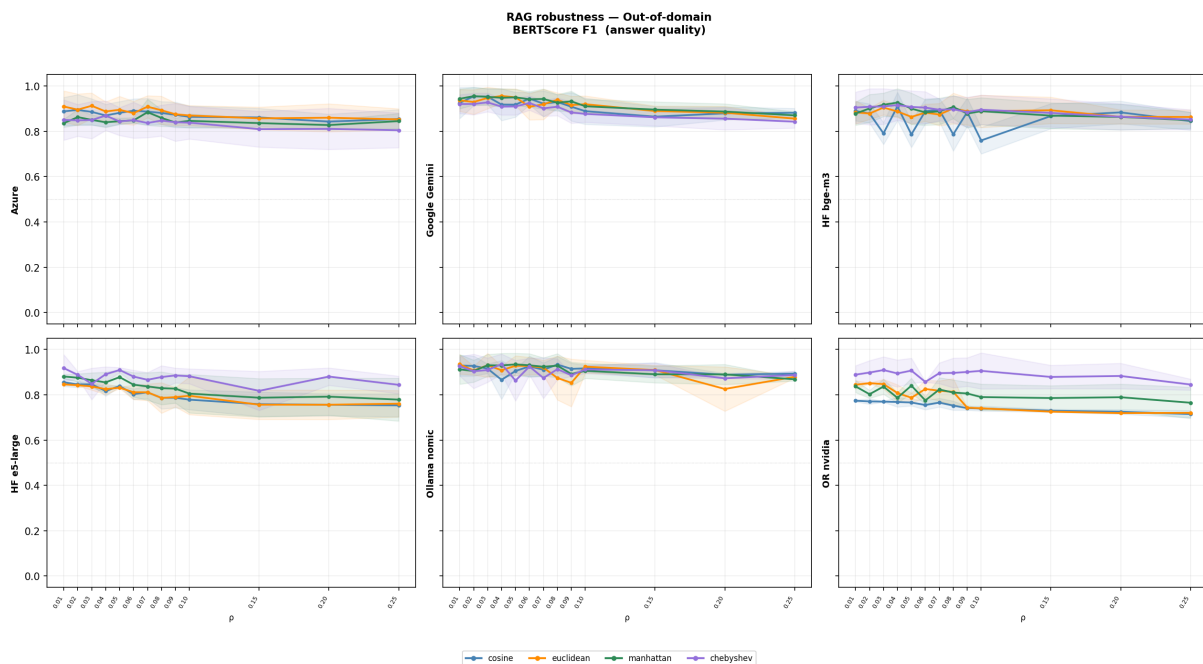


Figure 4.19. Out-of-domain answer quality (BERTScore F1) by provider. Curves and bands as in Figure 4.16.

Closing the three-level hierarchy.

The results of this section complete the three-level hierarchy of the thesis. The central empirical claim is the *retrieval/generation dissociation*: retrieval degrades sharply under moderate noise, while generated answer quality is substantially more robust.

A secondary claim, established by the provider stratification, is that embedding stability (as measured by NED and distortion) and retrieval stability are related but non-equivalent: Azure performs poorly on distortion yet leads on in-domain Jaccard, because the retrieval task rewards local neighbourhood preservation rather than global geometric uniformity. Both findings carry direct practical implications for the design of noise-resilient RAG systems, which are discussed in Conclusion.

4.4 Summary

This chapter has reported the empirical results of the three experimental modules, tracing typographical noise from the character-level confusion channel through embedding displacement to end-to-end RAG degradation. The key findings are consolidated in Table 4.6, whose embedding-stability columns are evaluated at the reference budget $\rho = 0.08$ and whose RAG column reflects in-domain cosine Jaccard, the most discriminating end-to-end criterion.

Table 4.6. Chapter summary across all three modules. Embedding columns at $\rho^* = 0.08$: $\hat{L}$ = mean cosine NED (lower better); $\bar{\kappa}_{\text{cos}}$ = mean cosine distortion; $\bar{\kappa}_{\text{euc}}$ = mean Euclidean distortion; ρ_{50}^{cos} = cosine displacement half-life (earlier = faster saturation). $t\text{-SNE}$: centroid chain regularity. RAG column: in-domain cosine Jaccard at $\rho = 0.08$ (higher better). ✓ best, ~ intermediate, ✗ worst.

Provider	$\hat{L}$	$\bar{\kappa}_{\text{cos}}$	$\bar{\kappa}_{\text{euc}}$	ρ_{50}^{cos}	$t\text{-SNE}$	RAG	Overall
HF_e5	✓	~	✓	~	✓	~	Best
Gemini	~	✓	✓	✓	~	~	Best
Azure	✓	~	✓	✗	~	✓	Good
OR_nvidia	✗	~	✓	~	~	✓	Moderate
Ollama	~	✗	✗	~	✗	✗	Poor
HF_bge	~	✗	✗	~	✗	✗	Poor

- **Noise channel (Section 4.1).** The fitted confusion matrix is dominated by phonetically motivated substitutions ($\mathbf{z} \rightarrow \mathbf{s}$, $\mathbf{j} \rightarrow \mathbf{g}$) and by space-deletion ($\chi(-) = 0.914$), but the characters with the greatest cumulative impact on embedding displacement are the high-frequency letters **e**, **t**, **h**, **o**, and **a**, which combine moderate confusability with ubiquitous presence in the query set. The chain mixes rapidly ($t_{\text{mix}} \approx 3,000$ steps) and has a well-defined stationary distribution, validating it as a controlled, reproducible noise source. Human typographical error is thus governed by how words sound rather than by the arrangement of the keyboard, and the letters that break a query are the common ones rather than the rare. The most fragile query in the set, “*what is a fandango*”, is fragile precisely because it is short and composed almost entirely of high-frequency characters, its meaning resting on a single content word that any edit is statistically likely to strike.
- **Embedding stability (Section 4.2).** Displacement saturates early under every metric and is front-loaded into the first few edits, so mean NED necessarily decreases with ρ : most of the damage a typographical error can do is done by the first one, and effort spent handling severely corrupted queries yields poor returns once that first error has already occurred. HF_e5 and Gemini occupy the top tier by the same criterion, low distortion, reached by two different routes: HF_e5 achieves the

lowest mean cosine NED (0.031 at $\rho = 0.08$) together with low Euclidean distortion ($\bar{\kappa}_{\text{euc}} = 2.69$), keeping displacement small in absolute terms throughout; Gemini instead has the lowest distortion across all four distances ($\bar{\kappa}_{\text{cos}} = 2.79$, $\bar{\kappa}_{\text{euc}} = 1.65$) and the earliest cosine saturation ($\rho_{50} = 0.030$), front-loading 77% of its total NED drop into the first three budget steps, so that displacement is larger initially but converges to the same narrow, query-independent band. What matters for the user is that both fail *uniformly*: every query is displaced by a comparable amount, so degradation is gradual and predictable rather than catastrophic for any single user. Azure is the instructive counterexample, and the true point of contrast: it is globally competitive in mean NED (0.077) yet exhibits the highest global cosine distortion ($\kappa_{\text{cos}} = 105$) and the latest cosine saturation ($\rho_{50} = 0.15$), concealing a $105\times$ spread between its most and least adversarial query-perturbation pairs. Azure fails *selectively*: the average user notices nothing, while a small minority find the system has effectively stopped working for them. Short queries are, across every provider, far more exposed than long ones, so spell-checking the query before it reaches the retriever addresses the regime in which nearly all the damage occurs, and is worth more than any choice among these six models; where that is not possible, the choice between a uniform failure mode (Gemini, HF_e5) and a selective one (Azure) should be dictated by whether the deployed query set is known and testable in advance, since a selective model is safe only under that condition.

- **Embedding space topology (Section 4.2).** The t-SNE projections add a structural dimension absent from the scalar metrics, distinguishing embedding spaces that *degrade* from those that *break*. The fragmented sub-cluster structure of Ollama ($\kappa_{\text{cos}} = 169$) allows perturbed queries to cross neighbourhood boundaries at modest budgets, producing the non-monotone centroid trajectories that are the geometric counterpart of its high distortion; HF_bge exhibits a related pathology via abrupt jumps at the single-to-double edit threshold ($\kappa_{\text{cos}} = 294$, the highest among all providers). Both pathologies manifest at the RAG level as catastrophic retrieval failures: the cosine Jaccard of Ollama collapses from 0.97 to 0.00 at $\rho = 0.02$, while HF_bge degrades more gradually but reaches only 0.26 at $\rho = 0.08$. A smooth embedding space degrades; a fragmented one breaks, ceasing to function at one additional character with no intermediate regime in which the failure might be detected and mitigated, and no downstream fix recovers a query once it has crossed into the collapse zone shared by all six providers. Azure’s isolated, low-density placement of ambiguous queries is the same geometric mechanism read as an asset rather than a liability: a query that sits alone retrieves unambiguously, which is precisely why Azure leads the in-domain RAG results below despite its poor global distortion.
- **Retrieval/generation dissociation (Section 4.3).** The most striking finding of the RAG module is that retrieval and answer quality degrade on entirely different scales: in-domain Jaccard collapses by 78% over the tested range while BERTScore drops by only 10%, and the two partitions produce nearly indistinguishable BERTScore curves despite very different Jaccard levels. The generator reconstructs semantically coherent answers from a partially or fully

replaced retrieved set, with corpus redundancy (parallel HTML and plain-text indexing) likely contributing to the robustness: a robustness that is borrowed rather than earned, and would not survive a sparser knowledge base in which each fact appears only once. The provider ranking is inverted between the embedding and retrieval levels: Azure, despite poor geometric uniformity, leads in-domain cosine Jaccard (0.87 at $\rho = 0.08$) because it places the specific deployed queries in stable, well-separated neighbourhoods, while HF_e5 and Gemini, strongest on the embedding metrics, trail by a factor of two on the task actually deployed. Distance-metric choice proves the single most consequential and least costly decision in the whole pipeline: cosine retrieval is markedly more stable than the ℓ_p norms at every budget, and a system currently retrieving under Euclidean or Chebyshev can roughly double its tolerance to typographical noise by switching metric alone, at no retraining cost. A RAG system may therefore lose most of its retrieval accuracy without the user noticing, so retrieval must be monitored on its own terms with a metric such as Jaccard, and an end-to-end answer score may even reward a system for failing reliably, as the out-of-domain Ollama result demonstrates.

Taken together, the three modules converge on a small set of actionable conclusions.

- First, because damage is front-loaded into the first typographical error, a lightweight spell-checking or normalisation layer ahead of the retriever addresses more of the problem than any choice among embedding providers.
- Second, no single provider is universally best: the choice is a genuine engineering decision between models that fail *uniformly* (Gemini, HF_e5), appropriate where the query distribution is open and unmonitored, and models that fail *selectively* (Azure), appropriate only where the deployed query set is known and can be tested in advance; Ollama and HF_bge, which combine high distortion with fragmented latent topology, are not defensible in either regime for noisy input.
- Third, retrieval should be evaluated, and configured, independently of generation: cosine distance should be the default choice for retrieval regardless of embedding model, and Jaccard-style retrieval monitoring is necessary because end-to-end answer quality can mask a near-total retrieval failure.
- Finally, aggregate or global statistics must always be read alongside their per-query counterparts, since a provider judged excellent on average, or catastrophic on average, may in fact be excellent on every query that will be deployed, or catastrophic on all but one, and only the query-level view reveals which.

Conclusion

This work began from a question posed simply in the introduction and left deliberately open: *when a user makes a typo, does the embedding model still understand them?* The answer that the preceding chapters support may be stated in one sentence, and everything that follows is a refinement of it: for a considerable range of ordinary human error the model does understand; beyond a certain point it does not; and the transition between the two regimes is neither gradual nor uniform across models.

The question was worth asking because the embedding model occupies a position in modern retrieval-augmented systems that is at once critical and largely unexamined. It is the first step through which a user’s intent enters the pipeline, and every subsequent stage is conditioned upon it: a system can answer only from what it retrieves, and it can retrieve only what the embedding model has placed near the query in its latent space [5]. Yet these models are trained, benchmarked, and selected almost exclusively on clean text, while the queries they receive in deployment are produced by people typing quickly, on mobile keyboards, frequently in a second language, and without proofreading. Prior work had established that performance degrades under such conditions, that strong rerankers cannot fully compensate for failures at the retrieval stage [8], and that small character-level perturbations can trigger large and unpredictable shifts in cosine distance [9]. What was missing was a principled means of quantifying that degradation, one whose conclusions would transfer beyond the particular models, queries, and noise levels tested in any single study.

Supplying such a means is the methodological contribution of this thesis, and it rests on a reformulation of the problem in metric terms. An embedding model is treated not as a black box to be benchmarked but as a map between two metric spaces: the space of strings under the Damerau-Levenshtein distance, and the embedding space under a choice of norm. Robustness thereby ceases to be a score and becomes a Lipschitz constant. The Normalised Embedding Displacement (NED) estimates that constant empirically, as the ratio of output displacement to input edit distance, and reports how far a typical error moves a typical query. Its bi-Lipschitz refinement supplies what an average cannot: the distortion index κ , dimensionless and therefore comparable across models of differing embedding dimension, records whether a model displaces all queries alike or reserves its worst behaviour for a few, while the displacement thresholds locate the noise level at which the displacement curve ceases to rise. The typographical noise itself was not assumed but modelled, as a character-level Markov chain estimated from two corpora of attested human spelling errors and validated spectrally, so that the perturbations injected are the

perturbations people actually produce. The values these indices assume are reported in Chapter 4; what follows is an account of what they reveal.

The most consequential finding is also the most general, and it holds across every model and every distance function examined. Robustness is not a slope but a plateau followed by an edge. Up to one or two errors in a query of ordinary length, all six models behave well: the perturbed query lands close to where the clean one would have, and retrieval succeeds. Once roughly a quarter of the characters have been corrupted, all six have failed, and failed identically, by collapsing the query into a dense region of the latent space where sufficiently corrupted queries converge irrespective of their original content. This upper boundary appears to be a structural property of the latent manifold rather than a defect of any particular architecture: beyond it the query no longer carries sufficient signal to be represented, and no subsequent processing recovers it. Between these two limits lies the regime in which the choice of model, the choice of retrieval metric, and the presence of a query-correction layer make a measurable difference, and it is also the regime in which real users operate. The engineering effort, in consequence, belongs at the edge and not on the plateau.

Within that intermediate regime, displacement proves heavily front-loaded: a user who mistypes a single character has already displaced their query a substantial fraction of the distance a thoroughly corrupted version would travel, and each further error contributes comparatively little. The practical corollary is inexpensive to act upon. Effort devoted to handling severely degraded queries yields poor returns, since by that point the query has already entered the collapse region, whereas a modest spell-checking or normalisation layer placed before the retriever addresses the regime in which nearly all of the damage is actually inflicted. The same asymmetry explains why the general resilience reported above is not distributed evenly across queries. A model that comfortably absorbs a typographical error in a full sentence may be defeated by the same error in a three-word query, where a single misplaced character can corrupt the only token bearing semantic weight and there are no others among which to dilute it. Robustness, in other words, is a property of the query as much as of the model, and systems whose users type terse search-box queries are materially more exposed than those whose users type complete questions.

The choice of retrieval distance function emerged as a larger determinant of end-to-end robustness than the choice of model, and it is the least costly to change. Retrieval under cosine distance proved substantially more resilient to typographical noise than retrieval under any of the ℓ_p norms, and dramatically more so than under ℓ_∞ , which responds to a single anomalous coordinate and will reorder the entire neighbour list on that basis alone. This is a matter of one line of configuration in any vector database. Under the ℓ_p norms, moreover, the differences between the six models largely disappear, so that a system retrieving with Euclidean distance derives little benefit from a premium embedding model: the metric has flattened away the very distinctions for which one is paying.

As to the models themselves, the evidence sorts them not into successes and failures but into distinct temperaments, each suited to different conditions. Some are uniform in their response, displacing every query by a comparable amount, so that as input quality deteriorates all queries degrade a little together and none collapses abruptly; Gemini and HF_e5 exhibit this profile, which is the appropriate choice where the query distribution

is open and cannot be inspected in advance, and where predictability is worth more than peak accuracy. Others are selective, performing excellently on the majority of queries while proving disproportionately fragile on a few. Azure is the clearest and most instructive instance, since its markedly high distortion index never in fact bit in deployment: on the closed documentation corpus it was the most stable retriever of the six by a wide margin, because it situates distinctive queries in well-separated regions of its space, and none of the deployed queries happened to fall among its adversarial cases. A selective model is an excellent choice where the query set is known and can be tested, and a hazardous one where it is not. Others still are brittle: their latent space fragments into dense sub-clusters separated by gaps, and a query displaced across such a gap by a single additional character arrives in an unrelated neighbourhood with no intermediate stage. Ollama and HF_bge display this pathology, and its consequences are stark: Ollama retrieves the correct documents for a clean query and none of them, not merely fewer, for the same query with one letter out of place. This is not degradation that can be accommodated by tuning; it is a discontinuity, and it occurs at the second error. Such models remain entirely serviceable where queries arrive validated or are generated programmatically, and are unsuitable in front of a human typist. The general principle, which no aggregate benchmark can convey, is that a model should be evaluated against the query distribution it will actually encounter: a poor distortion score is not a verdict but a warning that some queries will fare considerably worse than the average suggests.

The final finding concerns evaluation rather than construction, and has the widest implications. Retrieval and answer quality degrade on entirely different scales. Under moderate perturbation the retriever loses the large majority of the documents it ought to have found, while the answer ultimately generated remains close to the answer the clean query would have produced: the language model, presented with a partially incorrect context, generally reconstructs something serviceable from the fragments that survive. From the user's perspective this is a favourable result. From the engineer's it is a caution, for it means that retrieval quality is very nearly invisible from the outside; a team monitoring answer quality alone would observe a modest decline and conclude that the system was performing acceptably, while the layer beneath it had quietly lost most of its accuracy. The most striking illustration is a model that achieved the highest out-of-domain answer similarity of all six precisely *because* its retrieval had collapsed entirely: presented with irrelevant context, the generator reliably reported that it could find nothing, and that report matched the clean-query report almost exactly. An end-to-end score can, in short, reward a system for failing consistently. Retrieval must accordingly be measured on its own terms and never inferred from the answers it yields.

The framework opens two lines of inquiry that the present study leaves open, and the first of them is the more interesting. The noise channel employed here pools two categories of human error whose origins are quite distinct. *Cognitive* errors are phonetic and orthographic: a writer uncertain of a spelling substitutes a letter that sounds correct, rendering *which* as *wich*. *Mechanical* errors are motor: a finger strikes an adjacent key, two keystrokes arrive in the wrong order, a space is omitted. Both were deliberately combined into a single channel here, in order to reproduce the aggregate error profile of a realistic user population, and the picture obtained is accordingly a

blend of the two. There is, however, no reason to expect an embedding model to respond identically to them. A phonetic substitution tends to preserve the length and shape of a word and may leave its tokenisation intact, whereas a transposition or a lost space can re-segment the string entirely and present the encoder with a wholly different token sequence. Estimating the two channels separately and measuring NED against each would answer a question of direct practical consequence, since the two populations are distinguishable in the field: a user typing on a mobile device produces predominantly mechanical errors, while a user writing in a second language produces predominantly cognitive ones. Should one channel prove substantially more damaging than the other, robustness would cease to be a single figure attached to a model and become a property to be matched against a deployment context. The second line of inquiry concerns the passage from measurement to improvement. The NED framework was constructed to evaluate, but it supplies precisely the signal a training objective would require: a contrastive loss penalising large embedding displacement for small edit distance could be incorporated into the existing fine-tuning procedure for sentence encoders, with the bi-Lipschitz band furnishing the gradient directly. The sub-cluster fragmentation identified in Ollama and HF_bge is a particularly inviting target, since it remains unclear whether it originates in the training corpus, the pooling strategy, or the contrastive objective, and each hypothesis suggests a different remedy. The same indices extend, finally, to agentic pipelines, in which a corrupted query at the first retrieval step propagates through every subsequent reasoning iteration [37].

The introduction observed that research on embedding models has concentrated on clean, well-formed queries, while the queries that real people type are nothing of the sort. This thesis has rendered that discrepancy measurable. Robustness now possesses a name, a scale, and an interpretation: the NED index and its bi-Lipschitz band report how far a typographical error displaces a query, the distortion index reports whether a model distributes that harm evenly or concentrates it upon the unfortunate few, and the saturation thresholds report how much corruption a query can absorb before it ceases to be a query at all. The picture that emerges is, on balance, an encouraging one. Modern embedding models prove more resilient than one might have feared: within the range of error that ordinary users actually produce, most of them retain the meaning most of the time. Where they fail, moreover, the failures are not mysterious but structural, and structure admits of remedy. A retrieval metric can be changed in a single line. A spell-checker can be placed before a retriever. A model can be selected against the queries it will genuinely receive rather than against a leaderboard. None of these measures demands a new architecture; each demands only knowing where to look, which is what a measure is for. In a world where every interaction between user and system is mediated by text, robustness to human typing errors is not merely a desirable feature: it is a fundamental requirement.

