Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Electrical, Electronics and Communications Engineering
(36^{th}cycle)

Generative Approaches to Sound-Squatting: AI Tools and Validation

By

Rodolfo Vieira Valentim

******

Supervisor(s):
Prof. Marco Mellia, Supervisor
Prof. Idilio Drago, Co-Supervisor

Doctoral Examination Committee:
Prof. Tanja Zseby, Referee, Technische Universität Wien
Prof. Sebastian Garcia, Referee, Czech Technical University in Prague

Politecnico di Torino
2024

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Rodolfo Vieira Valentim
2024

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

I dedicate this thesis to my loving parents, my amazing girlfriend, my intelligent sister, and to my dearly missed mother-in-law.

Acknowledgements

The research leading to these results has been funded by SmartData@PoliTO center for Big Data technologies.

Abstract

Cyber-squatting, a cyber-crime involving the registration of domain names to exploit established trademarks or identities, encompasses various strategies, including sound-squatting. Sound-squatting leverages phonetic similarities to deceive users, a risk amplified by the proliferation of smart speakers, voice assistants, and audio content. Sound-squatting presents challenges due to the inherent variations in pronunciation across different languages and among individuals. This thesis explores the complexities of sound-squatting across single-language, and cross-language scenarios.

Our primary goal is to develop a robust methodology for generating sound-squatting candidates that effectively handles various linguistic scenarios. By employing a data-driven approach using machine learning models, particularly Transformer Neural Networks, we successfully produce sound-squatting candidates across diverse scenarios, surpassing traditional list-based models in capturing pronunciation nuances.

The research follows a multi-stage process, starting with a naive baseline model and advancing to sophisticated architectures utilizing raw audio, spectrograms, and token-based pronunciation encoding. Evaluations are both quantitative and qualitative, assessing homophone coverage, quasi-homophone generation quality, and multi-language support.

Furthermore, the thesis examines whether these models can predict user transcription errors resulting from pronunciation misunderstandings. We compare collected user data on transcription mistakes with the model outputs to evaluate predictive accuracy.

Practical implications are explored through case studies on domain registrations and Python package repositories. We analyze Transport Layer Security (TLS) certifi-

cates and the Python Package Index (PyPI) to identify sound-squatting candidates, revealing real-world exploitation potential.

Key contributions of this thesis include developing a comprehensive approach to sound-squatting generation using advanced machine learning techniques, providing a systematic validation framework for assessing these tools, and conducting an analysis of the tools' predictive capabilities regarding user transcription errors. Additionally, the thesis offers an extensive checking of the sound-squatting attack surface in domain registrations and Python packages.

This research enhances the understanding of sound-squatting and provides practical tools and methodologies to mitigate associated risks. The findings highlight the need for proactive cybersecurity measures and offer valuable insights for future studies and applications.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cyber-squatting is a cyber-crime in which the offender purchases or registers a domain name closely resembling an existing one, with the aim of profiting from recognizable trademarks, company names, or personal identities. It is applied in various contexts, including fake domains [28], phishing campaigns [53, 52], the hijacking of smart speakers [32, 74] and brand abuse. Several cyber-squatting strategies have been demonstrated in practice, including simple/frequent typos [71, 57, 4, 29], visual similarity between characters [28], and combination of common words [31, 37] leading to different attacks as typo-squatting, bit-squatting, homograph-squatting, combo-squatting [73], skill-squatting [32] and sound-squatting [43].

Sound-squatting is relatively a more recent technique that exploits pronunciations similar to legitimate names or brands to deceive users. Its importance is growing with the rise of smart speakers and voice assistants [32], as well as the resurgence of audio-exclusive content consumption, such as podcasts. Websites or products verbally advertised and misunderstood by users, or careless voice searches [11] can lead users to malicious content.

1.1 Linguistic Scenarios for Sound-squatting

Sound-squatting presents challenges due to the inherent variations in pronunciation across different languages and among individuals [73]. This complexity is further increased when multiple languages are involved, expanding sound-squatting possibilities. For instance, individuals may encounter difficulties when writing, pronouncing,

or recognizing a word pronounced in a foreign language, a scenario referred to as a *cross-language* scenario. Previous studies [32, 43] have predominantly focused on English homophones, which are existing words with identical pronunciations. However, these studies have limitations in terms of coverage, as they do not consider non-existing words, words with similar pronunciation (quasi-homophones), and cross-language scenarios.

Our primary concern involves the discrepancy between the language in which the pronunciation occurs and the language in which the writing or understanding takes place. Existing literature typically assesses the problem of sound-squatting under the assumption that the reading and spelling take place in the same language. However, when they differ, as in the case of a French individual needing to write "ash", pronounced correctly in English, there arise multiple possibilities. For instance, the French word "Hache" (pronounced [aˈʃ]) means "axe", while the English word "Ash" (pronounced [aˈʃ]) refers to the residue from burning.

These two words, originating from distinct languages, are categorized as cross-language homophones due to their identical pronunciations. Quasi-homophones, on the other hand, are alternative words that sound similar to the original word and have different spellings. Example of a quasi-homophone in English is the pair of words "write" and "right". Although these words have different meanings and spellings, they sound very similar when spoken aloud, especially in rapid speech or in certain accents. Expanding upon this concept, quasi-homophones also covers non-existent words that maintain grammatical coherence.

This quasi-homophone concept also extends to other languages. For instance, consider the Spanish word "caro" (pronounced [ˈkaro])), which means "expensive". An English speaker might misinterpret this as "carrot" due to the similarity in pronunciation. Similarly, the Portuguese word "pão" (pronounced [ˈpɐ̃w]), meaning "bread", could be misheard by an English speaker as "pound".

Given the amount of possibilities in which homophones and quasi-homophones occurs, encountering sound-squatting candidates that deviate from strict grammar rules or introduce entirely new words seems prudent. Hence, for the purpose of our analysis, the term "quasi-homophones" encompasses two or more words, whether existing or not, that share similar pronunciations. This definition also includes quasi-homophones across different languages or cross-language quasi-homophones.

To categorize the "linguistic scenarios" for sound-squatting *generation*, we employ a taxonomy that divides the problem into two categories: single-language, and cross-language scenarios. In short, this taxonomy helps in systematically addressing the complexities involved in sound-squatting generation.

In the **single-language scenario**, the language of pronunciation aligns with the language of understanding or spelling. For example, an English word like "night" might be replaced with its homophone "knight". Tools can also be *multi-language* when they handle multiple single-language scenarios. This can involve having a specific model for each language or a single model capable of dealing with multiple languages.

In contrast, the **cross-language scenario** arises when the language of pronunciation differs from the language of understanding. In such cases, modeling the problem calls for learning patterns to accommodate subtle variations across different language contexts. An example would be a French speaker hearing the English word "beau" (beautiful) and transcribing it as "bow" due to the similar pronunciation but different language and meaning.

1.2 Sound-squatting Tools

Based on the general cyber-squatting literature, the most common recommended mitigation strategy for cyber-squatting involves proactively purchasing domains susceptible to squatting [9]. We assume that the same holds true for all types of squatting techniques. Designing a tool to facilitate this mitigation process in the context of pronunciation similarities requires some consideration regarding pronunciation encoding.

Two commonly used methods for generating candidates are list-based models and data-driven models. List-based models rely on predefined lists of known homophones and common replacements to create potential instances of sound-squatting. These methods involve replacing either entire input names or parts of them with their homophones, resulting in a modified written form with exact or similar pronunciation. For example, this might include replacing "for" with "four" or "4" in "forever" (i.e., "4ever"). On the other hand, data-driven models utilize machine learning algorithms

and rely on training data to learn patterns and generate alternative written forms for the names, without the imposition of fixed rules.

In Section 2.1, we discuss these tools in detail and reference relevant research on the topic. While multiple tools exist for cyber-squatting candidate generation, they mostly neglect sound-squatting or use limited list-based approaches. Therefore, we believe we can improve sound-squatting generation.

1.3 Goal, Approach and Research Questions

Our objective is to introduce a methodology for generating sound-squatting candidates that can effectively address variations across all linguistic scenarios of Section 1.1. This is a challenging goal, given the complexity of sound-squatting and the amount of possible variations in pronunciation encoding, linguistic scenarios and modeling approaches. We introduce and investigate data-driven approaches to test whether one could avoid the need for establishing fixed rules to account for all linguistic variations.

In light of our goal and chosen approach, the following research questions are addressed in this thesis:

1. How does data-driven machine learning facilitate generalization across diverse sound-squatting scenarios and linguistic contexts?

We introduce multiple sound-squatting generation tools to investigate whether and how data-driven machine learning helps generalization across diverse sound-squatting scenarios and linguistic contexts. To understand the impact of several design decisions, we follow a multi-step investigation. We begin with a naive model to establish a baseline. Following this, we design a model based on state-of-the-art Transformer Neural Networks, which is well suited for Natural Language Processing (NLP) tasks. Subsequently, we conduct ablation studies to identify the essential components necessary for effective sound-squatting generation.

In our model design, we consider various decisions regarding data modality and encoding, such as whether to use raw audio, spectrograms, token-based pronunciation encoding. Additionally, we explore strategies for generating multiple candidate outputs.

To validate our models, we use both quantitative and qualitative evaluation methods. Quantitatively, we assess known homophone coverage and evaluate the quality of quasi-homophone generation by examining pronunciation similarity. Qualitatively, we evaluate the models' ability to support multiple languages and cross-language scenarios.

Once we have established that our tools can effectively produce homophones and quasi-homophones across diverse linguistic scenarios, we face our second research question:

2. Can AI-based sound-squatting generation replicate or anticipate mistakes users make during transcription?

It is unclear whether mistakes made by users due to pronunciation misunderstandings result in the creation of homophones and quasi-homophones for a word. This is precisely the case where abuse may occur, and we aim to verify if the proposed tools can indeed anticipate the mistakes users commonly make.

To investigate that, we have developed a questionnaire designed to collect data on the transcription mistakes users make when transcribing domain names. Our hypothesis is that these users' mistakes will correspond to some extent with the candidates produced by our tools. By comparing the collected transcription errors with the outputs of our sound-squatting generation models, we assess the models' ability to replicate or predict these mistakes, and thus their potential to be applied in practical scenarios.

Last, once we confirm mistakes users perform and check the ability of our methodology to anticipate them, we move to a practical exploration on the potential abuse of sound-squatting in the wild, posing the following third research question:

3. Are sound-squatting candidates produced by our methodology found in practical use cases?

This question focuses on verifying the existence of sound-squatting candidates for popular target brands and cross-referencing them with existing resources. Our examination includes registered domain names and Python software packages, aiming to identify potential instances of misuse or suspicious cases.

1.4 Thesis Contributions

The contributions of this thesis are the following:

- **A methodology for sound-squatting generation.** We present a methodology for generating sound-squatting candidates that couples with linguistic scenarios. We perform several ablation studies to help understanding the impact of the pronunciation encoding and how much different post-processing affect the generation quality.
- **A validation methodology for sound-squatting tools.** We propose a validation methodology for assessing the single and cross-language coverage of the tools, as well as the quality of quasi-homophone generation by comparing pronunciations. By following this methodology we show that the proposed tools can produce a significant number of existing known homophones, and the pronunciation similarity of generated quasi-homophones is comparable to that of homophones.
- **Analysis of tool's capabilities to anticipate user mistakes:** We conduct a two-step analysis regarding the ability of our and third-party tools to replicate or anticipate mistakes due to pronunciation misunderstanding. Our tools exhibit higher precision and recall compared to third-party ones.
- **Study of sound-squatting attack surface in domain registration using issued TLS certificates:** We conduct a study of the sound-squatting attack surface in domain registration by checking the existence of candidates using around 900 million certificates collected via Certificate Transparency logs. Our findings indicate that approximately 16% of the generated candidates are present in at least one certificate, and around 95% of the target domains have at least one candidate with a valid TLS certificate.
- **Examination of the sound-squatting attack surface in Python Packages:** We conduct a study using the Python Package Index (PyPI) by verifying the existence of candidates over a period of more than 900 days in the package platform. Our findings reveal that 0.35% of the generated candidates correspond to registered packages, potentially squatting 951 popular packages.

1.5 Thesis Outline

This section outlines the structure and main content of the thesis.

Chapter 2 addresses the background and related work. It starts by defining various aspects of cyber-squatting and sound-squatting, exploring their definitions, mechanisms, and implications. Furthermore, it discusses the role of Transformers Neural Networks in addressing cybersecurity challenges and introduces fundamental concepts such as the International Phonetic Alphabet and Mel Spectrogram. The chapter also discuss existing tools for squatting generation and investigates the methodologies employed in sound-squatting generation.

Chapter 3 outlines the methodology adopted in the study, detailing the approach employed for sound-squatting generative tools and presenting a generic pipeline for the generation process. Additionally, it discusses the method used for collecting users' transcription mistakes.

Chapter 4 presents our proposed tools. It introduces the `Auto-Squatter`, a system for simple IPA translation, providing a detailed description of its architecture, dataset, and training methodology. Additionally, it discusses the `Sound-skwatter`, a system including audio inbound with IPA translation, along with the `Sound-squatter`, designed for multi-language sound-squatting generation. The chapter concludes with an exposition on the `X-Squatter`, capable of cross-language sound-squatting generation, including system architecture, dataset, and training details.

Chapter 5 validates the introduce tools by examining the user's transcription mistakes and comparing their coverage performance.

Chapter 6 explores two potential applications in PyPI repository and domain impersonation, supplemented with a manual qualitative analysis.

Finally, Chapter 7 discusses on the linguistic coverage and scalability of the proposed solutions, highlighting their strengths and limitations. It also offers conclusions, summarizing the findings and suggesting future work.

1.6 List of Publications

1. **X-squatter: AI Multilingual Generation of Cross-Language Sound-squatting.** Valentim, R., Drago, I., Trevisan, M., and Mellia, M. 2024. *ACM Transactions on Privacy and Security (TOPS)*. https://doi.org/10.1145/3663569

2. **URLGEN Toward Automatic URL Generation Using GANs.** Valentim, R., Drago, I., Trevisan, M., and Mellia, M. 2022. *IEEE Transactions on Network and Service Management*, 20(3), p.3734–3746. https://doi.org/10.1109/TNSM.2022.3225311

3. **LogPrécis: Unleashing Language Models for Automated Shell Log Analysis.** Boffa, M., Vassio, L., Giordano, D., Valentim, R., Mellia, M., Drago, I., Milan, G., Houidi, Z. B., Rossi, D. 2024 *Computers & Security.* https://doi.org/10.1016/j.cose.2024.103805

4. **Lost in Translation: AI-based Generator of Cross-Language Sound-squatting.** Valentim, R., Drago, I., Mellia, M., and Cerutti, F. 2023. *In 2023 IEEE European Symposium on Security and Privacy Workshops (EuroSPW)* (pp. 513–520). https://doi.ieeecomputersociety.org/10.1109/EuroSPW59978.2023.00063

5. **Augmenting Phishing Squatting Detection with GANs**. Valentim, R., Drago, I., Trevisan, M., Cerutti, F., and Mellia, M. 2021. *In Proceedings of the CoNEXT Student Workshop (pp. 3–4). Association for Computing Machinery.* https://doi.org/10.1145/3488658.3493787

6. **AI-based Sound-Squatting Attack Made Possible.** Valentim, R., Drago, I., Cerutti, F., and Mellia, M. 2022. *In 2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)* (pp. 448–453). https://doi.org/10.1109/EuroSPW55150.2022.00053

7. **Tracking Knowledge Propagation Across Wikipedia Languages.** Valentim, R., Comarela, G., Park, S., and Sáez-Trumper, D. 2021. *Proceedings of the International AAAI Conference on Web and Social Media,* 15(1), p.1046–1052. https://doi.org/10.48550/arXiv.2103.16613

The thesis is based on the following publications: **X-squatter: AI Multilingual Generation of Cross-Language Sound-squatting**, **Lost in Translation: AI-based**

Generator of Cross-Language Sound-squatting, and **AI-based Sound-Squatting Attack Made Possible**. The publications **Augmenting Phishing Squatting Detection with GANs** and **URLGEN - Toward Automatic URL Generation Using GANs** represent initial efforts in applying generative artificial intelligence to cybersecurity. They have shown the potential of applying GANs to cybersecurity scenarios and have inspired the development of our novel tools and methods for sound-squatting generation. However, the applications of GANs in the sound-squatting scenario has proven unsatisfactory and, as such, these alternatives are left out of the thesis. The publication **LogPrécis: Unleashing Language Models for Automated Shell Log Analysis** leverages my expertise in generative AI acquired in the development of the thesis to process honeypot logs, in a novel research direction which I am currently pursuing in collaboration with colleagues. The scope of this work is however orthogonal to the one of this thesis. Lastly, **Tracking Knowledge Propagation Across Wikipedia Languages** is a result of an internship focused on a dataset study, which is not directly related to the core topics of this thesis.

1.7 Open-source and Datasets

1. `Auto-Squatter`: https://github.com/rodolfovalentim/code-thesis/tree/main/auto_squatter
2. `Sound-skwatter`: https://github.com/rodolfovalentim/code-thesis/tree/main/sound_skwatter
3. `Sound-squatter`: https://github.com/rodolfovalentim/code-thesis/tree/main/sound_squatter
4. `X-Squatter`: https://github.com/rodolfovalentim/code-thesis/tree/main/x_squatter
5. *LogPrecis* https://huggingface.co/SmartDataPolito/logprecis
6. Tracking Knowledge Propagation Across Wikipedia Languages. Valentim, R. 2021 Zenodo. doi: 10.5281/zenodo.4433137.

Chapter 2

Background and Related Work

This chapter provides essential background information for the understanding of the problem and the employed technology. Additionally, it includes a literature review of generative tools that specifically address cybersecurity and cyber-squatting techniques.

2.1 Cyber-squatting and Sound-squatting

Cyber-squatting is a class of attack in which a malicious actor tries to impersonate a legitimate resource [8]. Cyber-squatting gained notoriety with the widespread deployment of domain-squatting, a type of attack in which attackers register fake domain names to divert traffic from popular websites. For example, an in-depth search of over 224 million DNS records in 2018 identified 657 k domain names likely impersonating 702 popular brands [60]. There are several cyber-squatting strategies been demonstrated in practice, including simple/frequent typos [71, 57, 4, 29], visual similarity between characters [28], and combination of common words [31, 37] leading to different attacks as typo-squatting [58], bit-squatting [18, 44], homograph-squatting, combo-squatting [73], skill-squatting [32] and sound-squatting [43].

The Mitre Att&ck's CAPEC-631 [9] defines sound-squatting uniquely in the context of domain-squatting, as an attack in which *"an adversary registers a domain name that sounds the same as a trusted domain, but has a different spelling"*. The CAPEC-631 also enumerates possible mitigation to sound-squatting: (i) the deployment of additional checks when resolving names in the DNS, together with

the authentication of servers, and (ii) the preventive purchase of domains that have potential for sound-squatting. In addition to these protections, the system can warn the user about possibly malicious use of a word. In all cases, the targets of sound-squatting (e.g., brands and/or brand security providers) must know the names attackers will use to impersonate their brands. The tools proposed in this work come precisely to solve this problem, generating candidate names that can be used to mitigate the problem proactively.

2.1.1 Cyber-squatting in the Wild

In [30], the authors investigate the involvement of Certificate Authorities (CAs) in the HTTPS phishing ecosystem. Their study focuses on various types of squatting, including combo-squatting, typo-squatting, and homograph-squatting, to assess whether insecure practices of CAs contribute to an increase in attacks. They emphasize the elevated risk posed by squatted domains in the TLS environment, where end-users might mistake them for legitimate sites due to their appearance as "valid" in browsers' address bars.

Other studies [24, 61, 50] longitudinally assess domain squatting by employing tools to generate candidates and verify the existence of created domains. However, these works predominantly concentrate on techniques such as typo-squatting, combo-squatting, and homograph-squatting, overlooking sound-squatting.

The Python Enhancement Proposal (PEP) 541 [25] explicitly opposes name squatting within the PyPI ecosystem to prevent the distribution of malware through potential user confusion regarding package names. A notable example of these concerns surfaced in November 2022 when Phylum[1] reported on PyPI attacks: Malicious actors utilized Python to create a JavaScript extension that substituted cryptocurrency addresses in clipboards with wallet addresses under the attackers' control. While documented attacks against the PyPI ecosystem primarily involve typosquatting, there have been instances of squatting that inadvertently impacted non-English speakers.[2]

Yacong et al. [27] present a comprehensive analysis of security threats within software registries like PyPI and NPM, addressing various vulnerabilities, including

[1] https://blog.phylum.io/phylum-discovers-another-attack-on-pypi/
[2] https://metrodore.fr/i-have-been-powned.html

typo-squatting. They find that approximately 81.0% and 88.1% of removed packages on PyPI and NPM, respectively, share name similarities with at least one other package, indicating potential examples of typo-squatting. Additionally, a significant 96.9% of the malicious packages officially identified by NPM have names resembling benign packages.

These incidents highlight the inherent risks associated with name squatting within platforms like PyPI. Sound-squatting may introduce additional complexities for moderators to detect, potentially allowing packages to evade surveillance.

2.1.2 Sound-squatting Generation

In sound-squatting, list-based models are commonly employed to generate candidates. These models utilize predefined lists of known homophones and common replacements to generate potential instances of sound-squatting. They replace either entire input names or parts of them with their homophones, resulting in a new written form of the name with exact or similar pronunciation.

Nikiforakis et al. [43] generated potential instances of sound-squatting from a static database of English homophones, focusing on domain sound-squatting. Their approach involves substituting words in domain names with homophones. The authors then present a comprehensive evaluation and categorization of these generated candidates.

Kumar et al. [32] uncover the *skill-squatting* attack, where the attacker leverages systematic errors on the understanding of homophones to route Alexa Smart Speaker users to malicious applications. They show that around 33% of the systematic errors in the speech-to-text system are due to homophones. Later, Zhang et al. [74] formalize the skill squatting attack, calling it Voice Squatting Attack (VSA) and Voice Masquerading Attack (VMA). They also evaluate the feasibility of such attacks by deploying a malicious skill, which has been invoked by 2 699 users in a month by Alexa's Speech Recognition engine.

2.2 Transformers Neural Network

Recent advances in AI and Natural Language Processing (NLP) have made it practical to automate text translation via sequence-to-sequence (seq2seq) mapping. The Transformer model [69] represents a step forward. Compared to other seq2seq models such as Recurrent Neural Network (RNN), the Transformer has the capability of working with large sequences while keeping more context. This is possible thanks to the use of the attention mechanism that learns the relationship between all elements in sequences, as shown in Figure 2.1. The attention mechanism is analogous to a retrieval system, where there is a Query (Q), a Key (K) and a Value (V). Roughly, Q represents the word and the K:V represents the memory. The attention works like a database system where we query the memory, compare our query with a set of keys and get the corresponding values. In a Transformer, the attention mechanism is applied to the input sequence of the translation, to the target language and to the ongoing merge of the source sequence with its translation.

Several Transformer variants exist, not all restricted to NLP problems. Examples of algorithms include BERT [59] and GPT-2 [70]. Transformers have achieved the highest scores for Neural Machine Translation [69, 34] and their performance has represented an improvement for seq2seq problems.

The vanilla implementation of Transformers consists of an encoder and a decoder, each of which is a stack of N identical blocks. Each encoder includes a MultiHead Attention block and a feed-forward interconnected by *Add and Normalization* layers. The decoder is similar to the encoder; however, it adds to the encoder's architecture a cross-multihead attention mechanism that computes the attention between the input and the previous states of the target.

An important aspect to discuss regarding the Transformers is the difference between the training and the inference. During training, the Transformers use a mask over the target states given as input for the cross-attention head. This mask allows the model to learn multiple states simultaneously.

The inference process happens in steps where the decoder receives the previous states generated to compute the cross-attention. In this way, the output of the next element depends on the sequence of the previously generated elements. This is fundamental to correctly predict the next character in a word or the next word in a sentence.

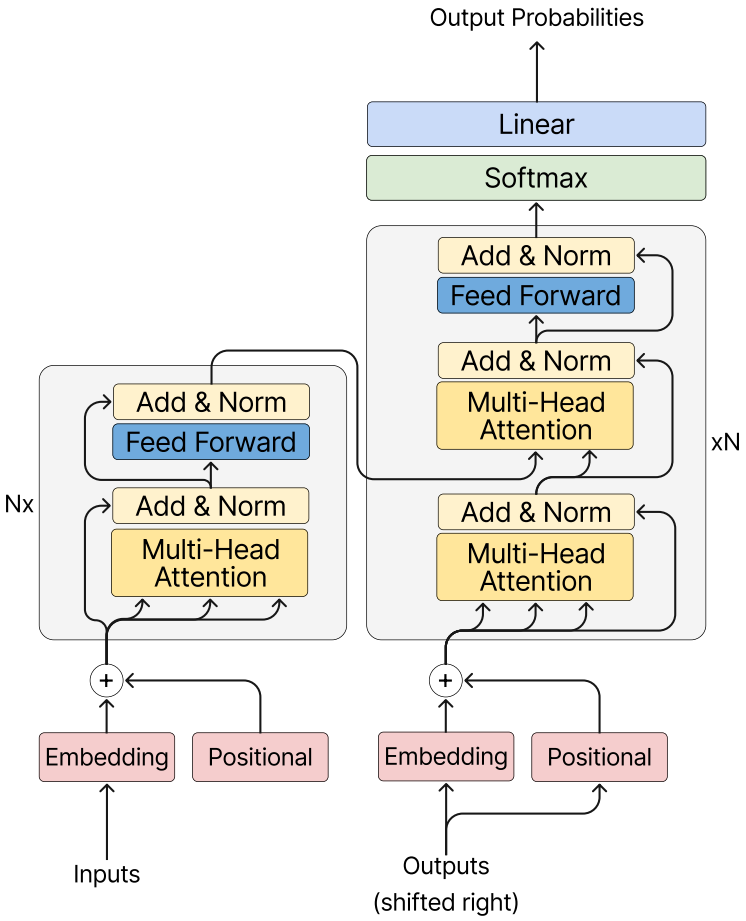


Fig. 2.1 Basic Transformer Neural Network architecture.

Throughout this thesis Transformers are used to generate words at the character level. It considers not only the most probable character but also explores possible words that may derive from considering the top-k most probable following characters, generating possible alternative ways to write the same input sequence.

2.3 Data Modality for Pronunciation

Continuous speech is perceived in distinct segments, classified based on how the vocal tract produces them, with each language utilizing its own set of segments [56]. Phonemes, the smallest linguistic units distinguishing word meanings, represent precise phonetic realizations often influenced by social factors and regional dialects [15].

2.3.1 Phonetic Alphabets

There are multiple ways to encode pronunciation for the usage in computational problems as Phonetic Alphabets, such as Speech Assessment Methods Phonetic Alphabet (SAMPA) [63], ARPAbet [16], and International Phonetic Alphabet (IPA) [6]. These phonetic alphabets are designed to represent the sounds of human speech in written form. SAMPA and ARPAbet are primarily used in computational linguistics and speech recognition systems, while IPA is more widely employed in linguistic studies and language teaching. SAMPA and ARPAbet use a combination of ASCII characters to represent phonemes, making them more computer-friendly, whereas IPA employs a broader range of symbols, including diacritics, to provide a more precise representation of speech sounds. Despite their differences, all three alphabets serve as valuable tools for accurately transcribing and analyzing the phonetics of spoken languages.

IPA serves as a standardized means of representing these phonemes with specific symbols. Widely adopted for representing pronunciation, IPA offers a consistent method maintained by the International Phonetic Association to transcribe sounds into written form. Each sound is assigned a unique character, allowing IPA to convey intonation and other linguistic properties effectively. While the IPA alphabet has evolved to reflect advancements in linguistics, it remains stable and representative, making it suitable for integration into Machine Learning applications. IPA utilizes

symbols from the Latin and Greek scripts, organizing them into categories such as vowels and pulmonic/non-pulmonic consonants.

In this thesis, IPA serves as the encoding mechanism for words into their respective pronunciations. Various solutions exist for translating words into IPA, such as the eSpeak NG (Next Generation) Text-to-Speech engine [23] and Epitran [41], which supports over 100 languages.

2.3.2 Articulatory Feature Vectors

Articulatory Feature Vectors, such as those found in resources like PHOIBLE [40], have an important role in linguistics and computational phonetics. These vectors encapsulate the acoustic properties of speech, offering a structured representation that facilitates comparative analyses across languages. By quantifying key Articulatory Features such as pitch, duration, and spectral characteristics, Articulatory Feature Vectors provide a standardized framework for phonetic research, enabling researchers to explore phonetic variation systematically. Moreover, they serve as valuable resources for computational models and tools, empowering natural language processing tasks such as speech recognition, synthesis, and dialect identification.

Projects like [40, 39, 22] aggregate IPA segments and associated features for different languages. Several proposals exist for creating Articulatory Feature Vectors that encompass features shared across languages. Implementations such as [33, 55, 42] convert IPA segments into these vectors, which are then employed in various Natural Language Processing (NLP) applications. PanPhon [42] is one of the most recent proposals mapping over 6 thousand IPA segments to 21 subsegmental articulatory features and it is used in `X-Squatter`.

2.3.3 Mel Spectrogram

The spectrogram depicts the magnitude and phase characteristics of a signal as it varies with frequency [51]. It provides a visual representation of an audio signal broken down into its constituent frequencies, with time represented along one axis and frequency along the other.

The Mel Spectrogram, a log-scaled variant of the Linear Spectrogram, is particularly favored for human perception due to its enhanced ability to differentiate

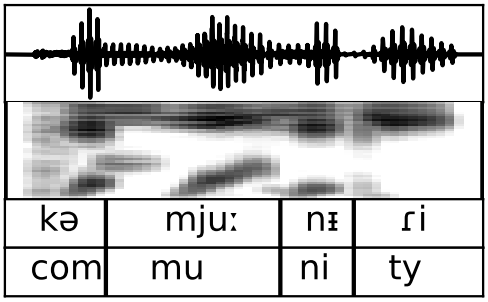


Fig. 2.2 Raw audio wave and its Mel Spectrogram aligned to the pronunciation and grapheme of the word *"community."*

between low-frequency sounds compared to high-frequency ones. This representation is commonly employed in machine learning applications, especially those related to speech, as it focuses on frequency components crucial for such tasks. Figure 2.2 illustrates the raw audio signal alongside its Mel Spectrogram, IPA pronunciation, and grapheme representation for the word *"community."*

In this thesis, the Mel Spectrogram serves multiple purposes: (i) learning the sound representation, which is subsequently used to generate graphemes with similar pronunciation to the original; (ii) assessing the similarity of generated squatting candidates based on their pronunciation; and (iii) aligning the pronunciation with the audio signal to estimate the duration of each phoneme, as demonstrated in Figure 2.2. Such duration estimation is crucial for spectrogram reconstruction, a topic explored further in subsequent sections.

2.4 Third-party Tools for Squatting Generation

There are several tools built to assist the process of proactively uncovering domains susceptible to squatting. These tools typically employs a variety of squatting techniques, such as typo, bit-flip, homograph, combo, and sound-squatting.

For clarity, we briefly list the key differences between sound-squatting and the other techniques. *Typo-squatting* primarily leverages small typos to create confusingly similar names (e.g., gogle). *Homograph-squatting* exploits characters from different scripts that visually resemble each other (e.g., g00gle). *Combo-squatting* leverages combinations of words to fool people into believing a resource

is legitimate (e.g., my-google). *Bit-flip squatting* relies on bit-flip errors that may change messages during transmission in insecure protocols.

Table 2.1 provides a succinct overview of the main tools employed for generating squatting candidates, along with the techniques each tool utilizes. These techniques include Typo, Homograph, Bit, and Sound squatting methodologies. The tools listed include `DomainFuzz` [21], which focuses solely on Typo squatting, while others like `URLCrazy` [36] and `URLInsane` [5] employ a broader spectrum of techniques, incorporating Typo, Homograph, Bit, and Sound methodologies. `dnstwist` [19] employs Typo, Homograph, and Bit techniques, while `AIL` [5] encompasses Typo, Homograph, Bit, and Sound squatting methodologies.

All these tools fall short in sound-squatting generation because they only consider exact homophones within the single-language scenario in English-US. For instance, AIL utilizes a list of homophones extracted from Wikipedia, last updated in April 2020[3].

Table 2.1 Tools for squatting candidate generation.

Tool	**Techniques**
`DomainFuzz` [21]	Typo
`dnstwist` [19]	Typo, Homograph, Bit
`URLCrazy` [36]	Typo, Homograph, Bit, Sound
`URLInsane` [5]	Typo, Homograph, Bit, Sound
`AIL` [5]	Typo, Homograph, Bit, Sound

[3] https://github.com/typosquatter/ail-typo-squatting/blob/main/ail_typo_squatting/generator/homophones.py

Chapter 3

Methodology for Sound-squatting Generation

This chapter outlines the methodology for generating sound-squatting candidates using generative artificial intelligence. The chapter (i) introduces a taxonomy for categorizing candidate-producing tools based on the type of data and linguistic scenario, (ii) elaborates on the architectures of our proposed tools to produce sound-squatting candidates, and (iii) discusses the methods used for collection users' data on transcription mistakes, which we utilize to validate our models.

3.1 General Pipeline

Our goal is to investigate the use of AI for sound-squatting generation. Therefore, we design a tool to assist in the generation process. This task requires careful consideration of pronunciation encoding, which can vary depending on the availability of data, computational resources, and the chosen generation approach.

Traditional tools (see Section 2.4) typically generate modified names and assess the availability of the name in the target resource (e.g., a domain name). Our focus, however, is solely on the generation of modified names. One of our research questions addresses the use of data-driven approaches to learn alternative spellings without relying on fixed rules. To achieve that we propose a pipeline composed of three main components: *Pronunciation Encoder*, *Data-driven Model*, and *Post*

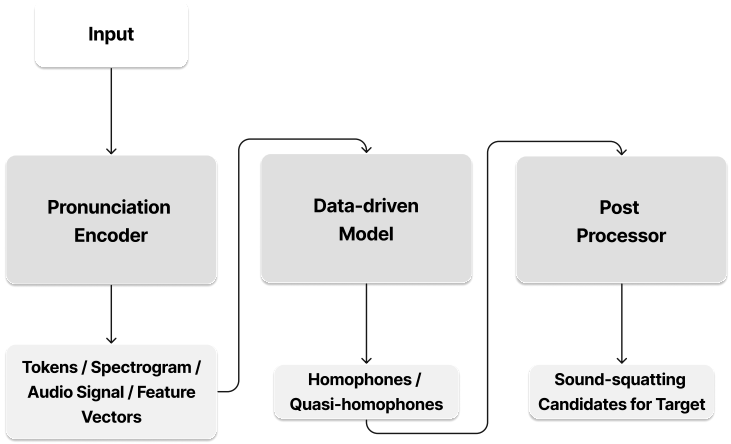


Fig. 3.1 Methodological pipeline for sound-squatting generation.

Processor. Figure 3.1 illustrates this pipeline, which aims to produce sound-squatting candidates for multiple linguistic scenarios.

The pipeline begins with an input that defines a target, such as a domain name. This is followed by the *Pronunciation Encoder*, which encodes the grapheme into its pronunciation. This step can vary depending on the model and may involve learning based on fixed rules (e.g., Grapheme to Phoneme tools) or even audio data. The resulting pronunciation can be represented as tokens, audio signal or other pronunciation representations. Subsequently, the *Data-driven Model* generates written forms (graphemes) for the input pronunciation. This model has the capability to produce alternative spellings, including homophones and quasi-homophones. Finally, the *Post Processor* transforms these into sound-squatting candidates, which may call for additional steps based on the application, such as the inclusion of top-level domains (TLDs) in domain squatting or the elimination of duplicates. Consequently, post-processing steps are applied as necessary.

3.1.1 Data Modality for Sound-squatting

Data-driven models leverage machine learning algorithms and require training data to learn patterns and produce alternative written forms for the names. The data can contain errors made by users, which the model can learn from by capturing the

distribution of these errors and replicating them in new data. Alternatively, the model can implicitly learn alternative spellings by grasping pronunciation or grapheme patterns.

While a model can learn from more complex data like audio signals or representations such as Spectrograms, this type of data is more expensive to acquire and their complexity results in more computational requirements to train the models. For these reasons, the selection of a data source and the modality must carefully consider this trade-off.

Regarding the data modality, data-driven models can be audio-based, token-based, or hybrid models. Where:

Audio-based models process the raw audio associated with words. They extract various audio features, such as pitch, intensity, and spectral characteristics to capture distinctive acoustic patterns. The model then learns how to alter some of these features to generate distortions that lead to the desired alternative written forms (e.g., the homophones). As said previously, these models suffer from the complexity of dealing with audio signals and controlling errors. In a way, they work similarly to a Speech-to-Text pipeline, where the generated text contains the sound-squatting candidates.

Token-based models operate on textual representations of names. They break down the input name into tokens, which can be either (i) normal grapheme tokens or (ii) pronunciation tokens. Pronunciation tokens rely on a phonetic representation of the words, for instance, using the IPA alphabet. These models learn to map the representation of the name back into a phonetically equivalent grapheme form, like homophones or quasi-homophones. However, token-based models can only generate sound-squatting candidates using pronunciation-spelling combinations that have been seen in the training data.

This is a limitation for cross-language scenarios, where different spellings lead to similar pronunciations that are encoded as different categorical tokens. For example, consider again the example of the Portuguese word “pão” (pronounced [ˈp ɐw]), meaning “bread”. It could be misheard by an English speaker as “pound” (pronounced [paʊnd]). Note that both the pronunciation and the spelling are different. This limitation highlights the challenge of covering all pronunciation variations and their corresponding written forms in such models.

Hybrid models use a pronunciation representations that is richer then token representation, but less expensive then audio signal or spectrograms. The advantage of such representation lies in its capacity to encapsulate phonetic similarities among phonemes. There are several proposals for this type of representation such as the articulatory vector representation [42], and the typological vector representation [33]. A key advantage of this approach is that it enables cross-language support. That is, it overcomes the problem created by the lack of particular mapping between graphemes and phonemes in pairs of languages. Indeed such representations allow one to obtain a list of *similar* phonemes in the target language, which is much harder with token-based representations (e.g., based on the IPA alphabet).

3.2 Taxonomy of Generative Models

Following the proposed methodology, we have introduced several alternative models to tackle the diverse scenarios and, as far as we are aware of, these have been the first efforts to introduce a data-driven methodology for sound-squatting generation [64, 67, 66].

Notably, we will introduce in the coming Chapter alternative models comprising:

1. **Simple IPA Translation** is a token-based single-language model we introduced in [64] trained to learn transliterations from IPA segments to graphemes for a given language. This model includes two modules: *IPA to English* and a *English to IPA*. One model transliterates the English input to IPA adding some variability based on a random sampling and this input it transliterated back to English-US, again adding a random noise. The output consists in a single homophone candidate. This approach represents a simple baseline for the problem.

2. **Audio Inbound with IPA Translation** is an audio-based single-language model (introduced in [65]), which operates similarly to the previous model but additionally learns features from audio, thereby incorporating similarities between IPA segments.

3. **Multi-language IPA Translation** is the first token-based multiple-language model for cyber-squatting generation, which we proposed in [67]. It features

the inclusion of special language tokens to generate homophones in multiple languages. However, being token-based, this model lacks the ability to generate homophones in cases where there are gaps in the representation, for instance, in cross-language squatting.

4. **Cross-language IPA Translation** is the hybrid model for cyber-squatting generation proposed in [66]. It features the use of Phonetic Feature Vectors to encode the IPA tokens allowing the generating of homophones in multiple languages as well as cross-language homophones.

These alternatives produce homophone and quasi-homophone that are later used as sound-squatting candidates. The implemented pipelines, linguistic scenarios and data-driven approach are detailed in Chapter 4.

3.3 Transcription Error Collection

Armed with the multiple alternatives to generate sound-squatting (including those found on previous works) we move to the question on whether these tools generate candidates that represent likely users' mistakes.

For that we perform a survey with users, who are exposed to a questionnaire. This questionnaire aims to gather data on mistakes made by users when transcribing domain names from audio. The transcriptions collected are intended to serve as validation for the tools, assessing their ability to anticipate user behavior accurately. To provide additional insights, we also collect user information such as language proficiency.

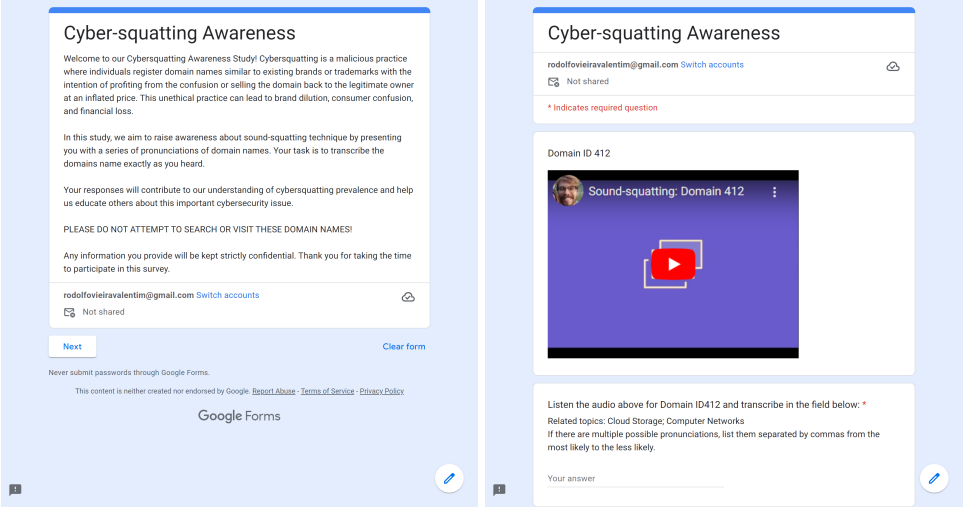
Participants are presented with pronunciations of existing domain names generated by the Google Text-to-Speech tool [3]. Subsequently, users are asked to transcribe the domain names based on their comprehension of the audio. The audio includes the pronunciation of the domain name and its Top-Level Domain. We configure the Text-to-Speech tool in English-GB and manually validate the pronunciation. While the pronunciation produced by the Google TTS tool is machine-generated, it closely resembles human pronunciation.

In this evaluation, we choose a controlled environment, inviting suitable volunteers from a few linguistic contexts. This approach helps us avoid the need for

additional data cleaning [38], which is often necessary when using crowd-source platforms like Amazon Mechanical Turk [1] or Clickworker [2], helping to improve the quality of the data. These platforms are typically considered for scaling the questionnaire.

The questionnaire contains 20 domain name pronunciations randomly selected from the Tranco Top 1 Million rank. All participants evaluate the same set of 20 random domains, following the methodology described in [54], which consists in a research aiming at understanding the effectiveness of typo-squatting. To random factor domain popularity, the selection of domains takes into account ranges of domain popularity when sampling: Initially, a set of domains is sampled from the interval spanning 1 to 1 000. Following this, subsequent intervals extended further, from 1 000 to 11 000, then from 11 000 to 111,000, and finally from 111 000 to 1 000 000. This selection process favors more popular domains over less popular ones.

Participants are presented with a Google Form (as depicted in Figure 3.2) containing contextual information at the beginning, followed by an explanation of the task. The target audience comprises individuals from different nationalities with varying levels of education. Participants are instructed not to check the domains, as this may pose a risk of visiting eventual squatted domains.



(a) (b)

Fig. 3.2 Google Form screenshot of the questionnaire that access user's mistakes during domain name transcription.

Additionally, contextual tags ("Related topics" in the picture above) related to the domain are presented alongside the audio pronunciation, enabling users to infer missing words if necessary. The tags are provided by Google Topics API [35]. We assume these hints mimics real-world communication scenarios, where context aids understanding. Each user can provide multiple possible transcriptions if needed. The domains, the contextual tags and the rank in Top 1 Million list are shown in Table 3.1.

Table 3.1 The domains, the contextual tags and the rank in Top 1 Million list used in the survey. The selection methodology favors more popular domains.

Tranco Rank	**Domains**	**Contextual Tags**
412	cloudns.net	Cloud Storage, Computer Networks
522	coinmarketcap.com	Investing, Business News
661	eastday.com	News
738	crunchyroll.com	TV and Video, Comics and Animation, Arts and Entertainment
741	americanexpress.com	Credit and Lending, Advertising and Marketing, Shopping, Banking
2 732	fsu.edu	Colleges and Universities
5 522	retailmenot.com	Coupons and Discount Offers
5 685	pdst.fm	Law and Government, Education
5 743	awsdns-18.com	Web Services, Comics and Animation, Online Communities
7 253	pornbox.com	Unknown
30 865	nanning.gov.cn	Business and Industrial, Law and Government
86 722	centrinvest.ru	Real Estate Services
87 700	cakecentral.com	Baked Goods, Cooking and Recipes
91 185	playerup.com	Arts and Entertainment
103 992	marnet.mk	Arts and Entertainment
118 015	sexymasseur.com	Unknown
474 837	bsta.rs	Accounting and Auditing, Tax Preparation and Planning
576 429	uir.ac.id	Colleges and Universities
741 359	kitchenwaresreview.com	Kitchen and Dining
964 077	informatica6.com	Software, Computers and Electronics

The data gathered from this questionnaire serves the purpose of elucidating whether sound-squatting generation tools possess the capacity to anticipate transcription errors resulting from user misunderstandings. This verification procedure consists on utilizing the tools to generate potential domain name variations for the designated domains. Subsequently, the data provided by users is compared with the synthetic data generated by the tools for comparative analysis.

Chapter 4

Our Proposed Models

This chapter presents our proposals to generate sound-squatting candidates using generative artificial intelligence. Building upon the methodological framework outlined in the preceding sections, this chapter delves deeper into the approach, describing details of the pipeline methodology and the developed tools.

At the core of our proposal is a data-driven methodology designed to generate sound-squatting candidates that are homophones and quasi-homophones of the input. Figure 3.1 provides a visual representation of the methodological pipeline, illustrating the steps involved in the generation process. From grapheme encoding to post-processing refinements, each stage of the pipeline is designed to produce candidates adequate to diverse sound-squatting scenarios.

Throughout this chapter, we provide a comprehensive overview of the proposed methodology, detailing the tools developed, the linguistic scenarios considered, the dataset and the training process employed.

4.1 `Auto-Squatter`: Simple IPA Translation

We first introduce our simplest model, `Auto-Squatter`. Initially presented at the 2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), this tool leverages two token-based single-language models. These models are specifically trained to generate transliterations from International Phonetic Alphabet (IPA) segments to English-US and, conversely, from English-US to IPA segments.

Notably, Auto-Squatter introduces variability into its outputs for identical inputs through the inclusion of random noise in the encoder's output. This simple idea serves as baseline and motivates our search for more robust methods able to cover more complex linguistic scenarios.

4.1.1 System Description

Figure 4.1 illustrates the tool pipeline, which employs a Transformer Neural Network to translate English-US words into their corresponding pronunciations in the International Phonetic Alphabet (IPA) and vice versa.

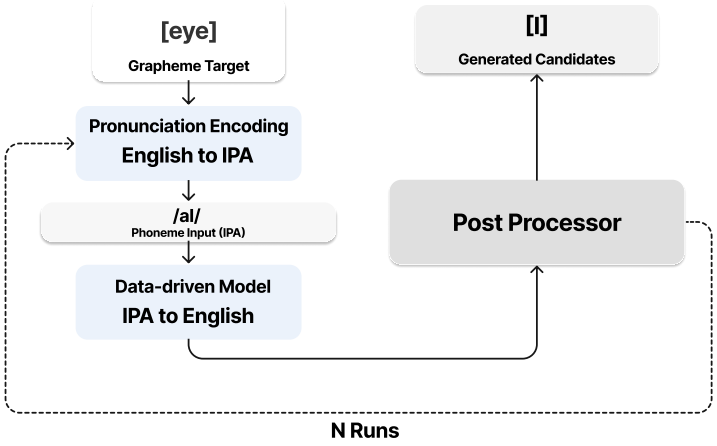


Fig. 4.1 Pipeline to generate homophones using Auto-Squatter. The *Post Processor* calls the model N times to generate multiple candidates.

In conventional translation tasks employing Transformers, output consistency is typically preferred (e.g., for the same input, the same output is expected). However, in the context of homophone generation, we search for models capable of generating multiple alternative outputs for the same input. These alternatives should closely resemble the expected output while introducing variations that do not compromise pronunciation, thus producing homophones and quasi-homophones of the input word.

To achieve this variability, each model is adapted to incorporate randomness by adding random noise sampled from a normal distribution to the latent representation

of the encoder's output. This addition is shown in Figure 4.2. This augmentation of the latent space confounds the decoder, resulting in outputs that maintain phonetic similarity to the originals, augmenting the amount of homophones generated for a given input word.

Generating multiple candidate outputs requires multiple model runs. For this reason, the *Post Processor* calls the model N times. As the noise varies with each iteration, slight changes in the output occur with each run. After numerous iterations, the candidates are selected from the unique samples produced by the pipeline.

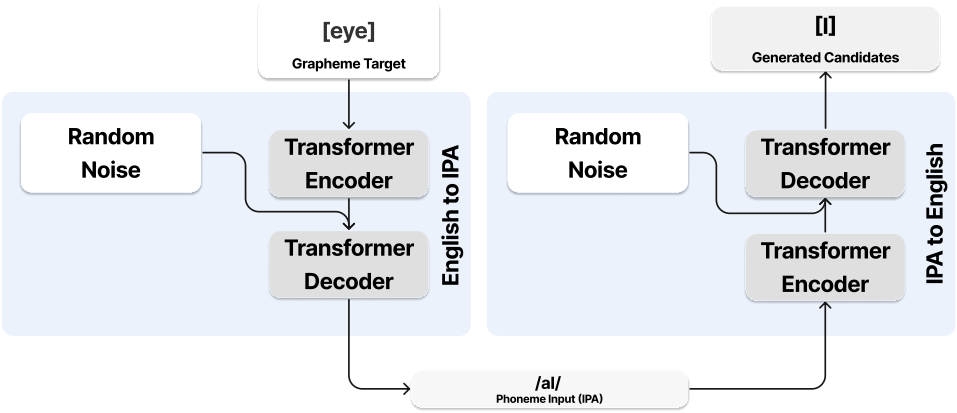
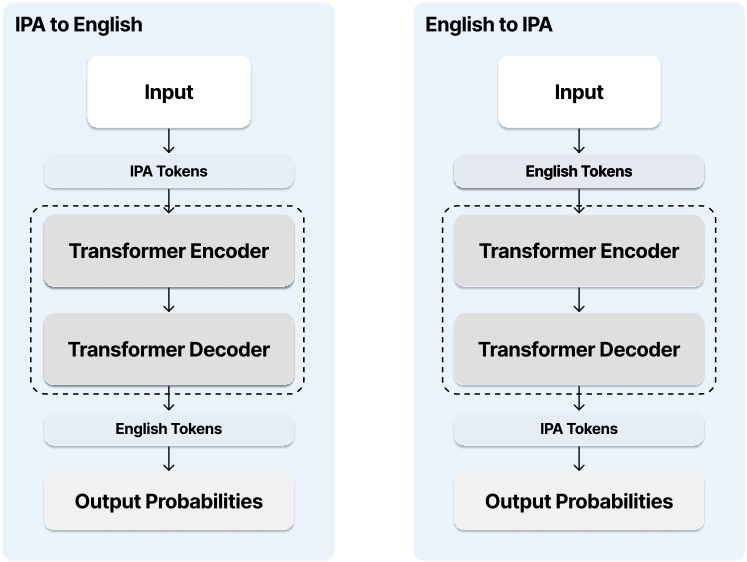


Fig. 4.2 Detailed pipeline to generate homophones using `Auto-Squatter`. The homophone generation pipeline incorporates a mechanism where random noise is introduced to each encoder output. This strategic addition of noise disrupts the model's processing, prompting it to generate alternative outputs.

4.1.2 Dataset and Training

To train this model we use a comprehensive collection of English-US words and their corresponding pronunciations, as detailed in [20]. This dataset contains a total of 125 923 word-pronunciation pairs.

During the training phase, the two translators are individually trained as shown in Figure 4.3. Specifically, for the English-US to IPA task and vice versa, one model is dedicated to converting English-US words to their IPA representations, while the other model focuses on the reverse task of converting IPA representations to English-US words.



(a) (b)

Fig. 4.3 The training process for Auto-Squatter involves a mirrored configuration for the models. This means that the output of one model serves as the input for the other model, and vice versa. (a) IPA to English; (b) English to IPA.

Both models within `Auto-Squatter` share identical architecture and hyperparameters. They are built upon the Transformer architecture featuring 2 heads, a latent dimension of 2048, and a sequence size of 25 tokens. Implementation is carried out using the Keras Framework [14], with training being performed locally.

During training, each model undergoes 10 epochs with a batch size of 64. The training dataset constitutes 80% of the samples, while 10% is allocated for validation and 10% for testing.

The selection of hyperparameters was guided by feature engineering considerations. The decision to employ a reduced number of heads, compared to other NLP setups, is justified by the observation that the translation from English to IPA does not need the model to learn long dependencies within the sequence.

The noise injection occurs during both the training and inference phases. Experiments conducted without noise resulted in output inconsistencies, since the noise during training affects the model convergence. The noise introduced is sampled from a normal distribution with a mean of 0.0 and standard deviation of 1.0.

4.1.3 Architectural Details

Figure 4.4 illustrates the process of adding noise to the Encoder's output. The Encoder's output is represented as a matrix of dimensions $B \times E \times S$, where B denotes the batch size, E represents the embedding size, and S indicates the sequence size.

`Auto-Squatter` hyperparameters are detailed in Table 4.1. These hyperparameters were selected based on best practices in Transformer architecture and training methodologies. As previously mentioned, the decision to reduce the number of heads from the standard Transformer configuration has been made to align with the task's complexity while also helping to reduce the overall model size.

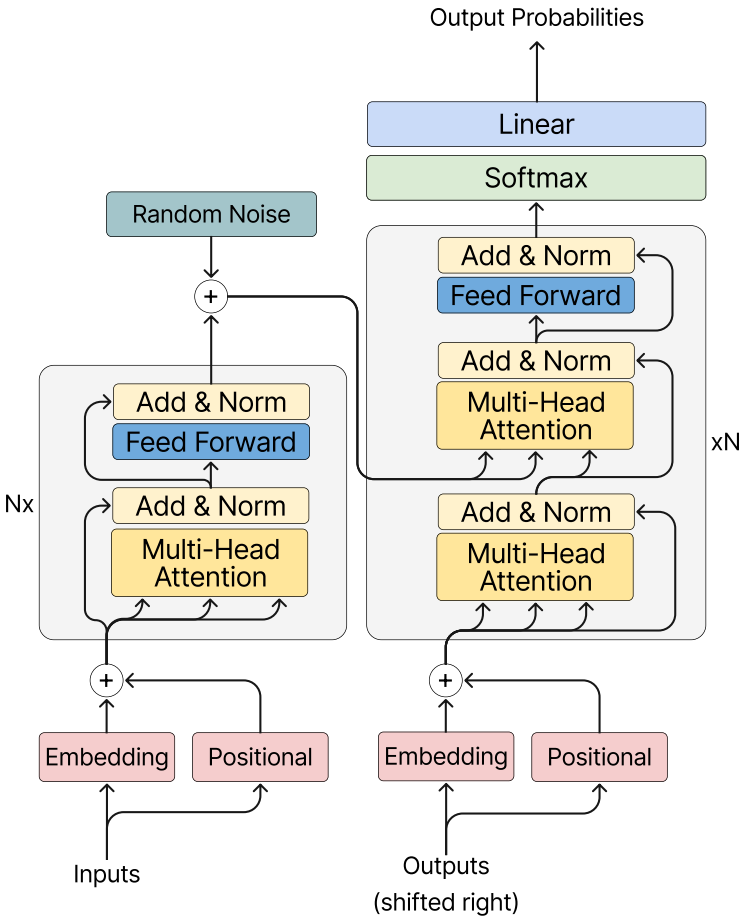


Fig. 4.4 Block diagram of a modified Transformers architecture showing noise insertion into the encoder output. This architecture is the same for both models in Auto-Squatter.

Table 4.1 `Auto-Squatter` Hyperparameters

Hyperparameter	**Value**
Number of heads	2
Latent dimension	2,048
Maximum Sequence Length	25
Epochs	10
Batch Size	64
Training dataset split	80%
Validation dataset split	10%
Testing dataset split	10%
Noise distribution	Normal
Noise mean	0.0
Noise standard deviation	1.0
Optimizer	RMSprop
Learning Rate	0.001
ρ	0.9
momentum	0.0
ε	10^{-7}
Training Epochs (Each Model)	10
Training Steps (Each Model)	1970

4.2 `Sound-skwatter`: Audio Inbound with IPA Translation

This section introduces `Sound-skwatter` a tool designed to automatically generate potential attack words using Transformer Neural Networks and acoustic models as feedback. `Sound-skwatter` generates candidates for any given target name, working at the sub-word level and allowing configurable approximations during the search for candidates. `Sound-skwatter` can be trained for any language. For training, the network receives as input (i) the International Phoneme Alphabet (IPA) representation of the word and (ii) the spectrogram extracted from the target word pronunciation audio signal. At inference, it recreates the written form (*grapheme*) while also considering pronunciation. `Sound-skwatter` uses a sequence-to-sequence task to find written alternatives with similar pronunciations.

The initial version of this tool [65] is discussed in the report titled "Sound-skwatter (Did You Mean: Sound-squatter?) AI-powered Generator for Phishing Prevention", available on the Arxiv.org platform, submitted on October 10, 2023.

4.2.1 System Description

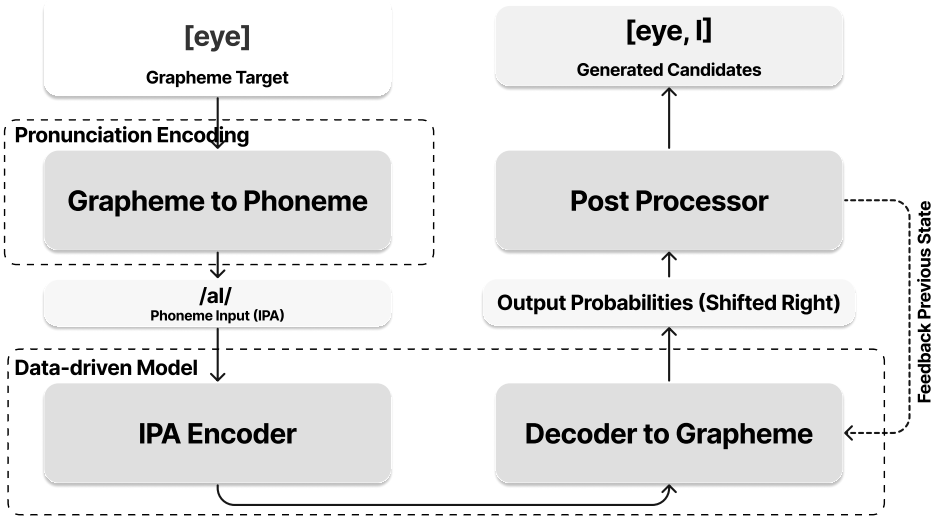


Fig. 4.5 The process to generate candidates is composed of an *Feature Vector Encoder* that maps the input to a latent space and a *Grapheme Decoder* that reconstructs the input.

`Sound-skwatter` is a tool capable of creating homophones which are later used as sound-squatting candidates which translates from phonemes to graphemes. The tool pipeline used for inference is describe in Figure 4.5. The generation pipeline is built upon four essential components (detailed in the following):

1. The ***Grapheme to Phoneme*** (G2P) language specific component converts the input word presented in grapheme format (in the example, the word "`eye`") into its corresponding representation in the International Phonetic Alphabet (IPA) (such as `/aɪ/` for British English).

2. The ***IPA Encoder*** component transforms the sequence IPA tokens in a latent space using a Transformer-based sequence-to-sequence model.

3. The ***Grapheme Decoder*** (P2G) component recursively decodes the latent representation into characters to compose a new grapheme form that presents similar pronunciation of the input word (such as "`I`"). The initial state fed to the model selects the target language.

4. The ***Post Processor*** component engages in beam search across the output logits and makes token selections according to probabilities derived from the decoder's output. This process is crucial for generating numerous quasi-homophones, which are alternative words that sound similar to the original word but have different spellings, all from a single pronunciation.

Next, each component is presented according to its role and contribution.

Grapheme to Phoneme (G2P)

The *Grapheme to Phoneme* component transforms written words into their respective IPA representations for a specific language. There are different tools available for that, including solutions that work for multiple languages. The most common approaches are either i) rule-based or ii) data-driven. Rule-based models employ predefined rules to convert word pronunciations based on their spelling. Data-driven models use machine learning techniques and annotated data to build a model for this task.

The previously presented `Auto-Squatter` uses a data-driven *Grapheme to Phoneme* model. However, for `Sound-skwatter`, the default option is the eSpeak NG (Next Generation) text-to-speech engine [23] and Epitran [41] for transliterating text into IPA. eSpeak NG performs better for English-GB and English-US.

The shift from a data-driven to a rule-based G2P model is due to convenience, as the available G2P tools provide sufficient quality to support the viability of our proposal. Additionally, the rule-based tools do not require additional training for every new language.

IPA Encoder and *Grapheme Decoder*

The fundamental components of `Sound-skwatter` consist of the *IPA Encoder* and *Grapheme Decoder*, which rely on the self-attention mechanism to learn the translation of a sequence of feature vectors into grapheme format. Architecture details can be found in Section 4.2.3.

When presented with a sequence of feature vectors obtained from the *IPA Encoder* model and predictions for each of the preceding $(N-1)$ characters, the

Grapheme Decoder module is responsible for estimating the probabilities associated with each potential character becoming the N-th character of the output. Afterward, the *Post Processor* component analyzes these probabilities and provides the historical context back to the *Grapheme Decoder* for generating the next forecast.

Post Processor: the Quasi-Homophone Generation

In the inference process, the *Post Processor* receives as input the probabilistic forecasts of the next character from the *Grapheme Decoder* and maintains a record of the prediction history to provide feedback to the *Grapheme Decoder*. Operating as an auto-regressive model during inference, more than one candidate quasi-homophone can be generated by adjusting the feedback history sent back by the *Post Processor*. A Beam Search `top-K` algorithm is used to identify the best candidates.

During each step of inference, the K most probable predictions of the *Grapheme Decoder* are stored by the *Post Processor*, and alternative histories are constructed to be fed back to the *Grapheme Decoder* in the subsequent step. Illustrated in Figure 4.6, this iterative process forms a tree (with $K = 2$), where each edge corresponds to an associated probability. In the example, starting with the IPA representation of the word `by`, each node represents the two most likely next characters given the preceding character sequence. The letter `b` emerges as the primary candidate for the first letter (with a probability of 99.72%), while `p` is the second-highest. Subsequently, `b` can be followed by `u` or `y`, and so forth. After four iterations, the process generates 15 different variations of writing `by`. Note that the generation process is stopped by the *Post Processor* when the *Grapheme Decoder* outputs the special character `EoS` (End of Sentence), as observed in the generation of `by`.

To find the best quasi-homophone candidates, *Post Processor* computes the joint probability of each leaf as being the product of the edge probabilities. At each step, *Post Processor* ranks current leaves by joint probabilities and keeps the L most likely ones to prune the search space and avoid pursuing branches with a low likelihood of producing good quasi-homophones. The number of iterations M at which stop the generation process, the number of candidate predictions (children) L to generate at each step, as well as the number of best candidates K to keep are additional hyper-parameters of `Sound-skwatter` which can be either defined manually or identified using standard local-search procedures.

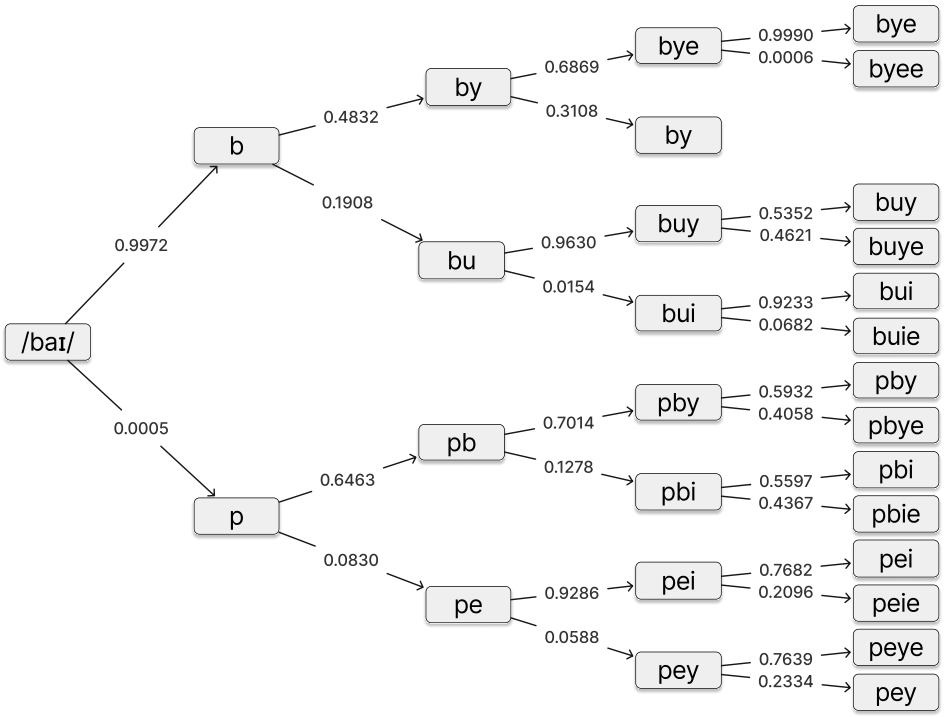


Fig. 4.6 Illustration of the inference process with $K = 2$ children per node. At each inference step, we collect the two next characters with the highest probability. The left and the right children have the second-highest and highest probability, respectively. Some nodes do not have children because they reach the "End of Sentence" state.

For the purpose of this work, given the length of the input word N, *Post Processor* iterates $M = N + 6$ times, generating potentially K^{N+6} candidates. To limit the exploration, we set $L = 64$ and choose $K = 2$. These parameters have been determined by manual inspection, and they are heavily dependent on the tasks and datasets of interest. These values were obtain empirically by looking at the pairs of words and pronunciations and by observing that no pronunciation is longer than the word by more than 5 tokens.

4.2.2 Dataset and Training

The architecture used to train the *IPA Encoder* and *Grapheme Decoder* is depicted in Figure 4.7. Both modules are trained jointly with one further function, *Decoder to Mel*, exploiting multi-modal data coordinated by the *Duration Predictor* function. The overall training task thus:

- is analogous to a sequence-to-sequence model, where the *IPA Encoder* and *Grapheme Decoder* together translate from IPA to the target language grapheme. At the same time the Mel Spectrogram supervisory signal is reconstructed from the phoneme vector representation and the forecast duration of each phoneme in the spectrogram domain;
- uses as input pairs $\langle x, \textit{to Mel}(\textit{Text-to-Speech}(x))\rangle$, where x is a word in phoneme format, and *to Mel* and *Text-to-Speech* are, respectively, one of the many existing software that generates a Mel Spectogram of audio signals and Text-to-Speech software, eventually first calling the *Grapheme to Phoneme* function for having it in IPA. The dual-modality input is required for generating quasi-homophones of good quality;
- outputs the word in grapheme format (via *Grapheme Decoder*) and the Mel Spectogram (via *Decoder to Mel*) from the vector representation which *IPA Encoder* outputs. The loss function used for training needs to carefully balance the training process between these two modalities. The loss is computed by summing the $L1$ and $L2$ Loss of the Mel Spectrogram reconstruction and the Cross-entropy Loss calculated between the predicted and the expected grapheme.

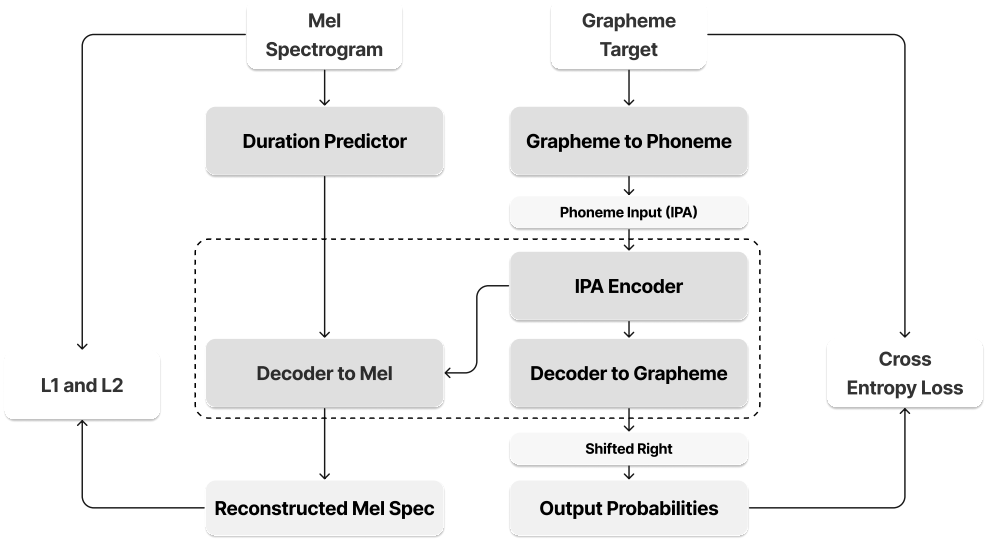


Fig. 4.7 The training architecture for learning how to generate quasi-homophones. The dotted box highlights the components that are trained for the generation of quasi-homophones: *Duration Predictor* is a pre-trained function. It receives as input the phoneme translation of a word and the duration of each phoneme in the expected spectrogram and outputs a reconstructed spectrogram and the probabilities that are used to find the grapheme translation.

Duration Predictor is a pre-trained function which coordinates the flow of multi-modal data by feeding the predicted temporal duration of each of the phonemes in *Grapheme to Phoneme*(x) to *Decoder to Mel*. It uses Convolution and LSTM Units to minimize the Connectionist Temporal Classification (CTC) loss [26], i.e., a loss between a continuous time-series the Mel Spectrogram and a target sequence the IPA word. The former is obtained by applying the Fast Fourier transform (FFT) over windowed segments of the audio signal and a transformation to Mel Scale. Given K the length of x, our word in phoneme format, *Duration Predictor* outputs $\langle d_1, d_2, \ldots, d_K \rangle$, where d_i represents the number of windowed segments of the audio signal predicted to be associated to the i-th phoneme of x.

English-US was chosen because it is the most widely used language for digital services and on the Internet [47]. Additionally, English-US exhibits phonetic inconsistency, making it a potential target for squatting. The training dataset contains English-US words sourced from the GNU Aspell [7] reference word list. This list, which forms the basis of GNU Aspell, is a free and open-source spell checker and contains a total of $125,929$ words in American English.

To acquire the pronunciation, the open-source software eSpeak NG Text-to-Speech is used. It supports more than 100 languages, and it is set for English-US to solve the grapheme-to-phoneme translation and to produce the speech sound signal. The generated speech sounds artificial to a human ear, but it is sufficient to provide the model with acoustic information about the pronunciation. The final step involves converting the sound signal to a spectrogram at the Mel Scale, a process carried out using the Torch Audio library [72].

The model is trained with a batch size of 64 words. As in the previous case, the training set comprises 80% of the samples, while 10% is allocated for validation and 10% for test sets. The Adam optimizer with $LR = 0.0001$, $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\varepsilon = 10^{-9}$ is employed. A step learning rate decay is scheduled with $\gamma = 0.1$ every 10 epochs. The model undergoes training for 30 epochs, equivalent to approximately $47k$ steps, requiring around 168 minutes on a single Nvidia Tesla V100. *Duration Predictor* undergoes training with the same dataset and the same Adam optimizer, converging at around $100k$ steps.

Table 4.2 `Sound-skwatter` Hyperparameters

Hyperparameter	**Value**
Number of heads	8
Latent dimension	512
Maximum Sequence Length	100
Epochs	30
Batch Size	64
Training dataset split	80%
Validation dataset split	10%
Testing dataset split	10%
Optimizer	Adam
Learning Rate	0.0001
ε	10^{-9}
Learning Rate Decay γ	0.1
Decay Frequency (Epochs)	10
Training Epochs (Model)	30
Training Steps (Model)	$\approx$ 47,000
Training Steps (*Duration Predictor*)	$\approx$ 100,000

Impact of acoustic feedback

In comparison with the previous proposal, `Sound-skwatter` includes audio features and the joint training with the Mel-spectrogram. To assess the impact of each aspect on results, a different training approach for `Sound-skwatter`, excluding the joint training with the Mel-spectrogram, is executed. The revised architecture is illustrated in Figure 4.8. In this alternative model, both the *Duration Predictor* and the *Decoder to Mel* components are omitted, resulting in the absence of $L1$ and $L2$ Loss terms for Mel Spectrogram reconstruction. The model without audio feedback essentially operates as a token-based model, performing a straightforward IPA translation in practice. Training for this model follows the same procedures and hyperparameters as the complete model. The pipeline remains the same.

4.2.3 Architectural Details

Figure 4.9a details the architecture of the model used for homophone generation. The diagram includes the *IPA Encoder*, *Grapheme Decoder* and *Decoder to Mel*. Figure 4.9b details the inside of the decoder block. Figure 4.9c is a high-level representation of the Length Regulator. The Length Regulator expands the *Feature Vector Encoder* output to the same order of magnitude as the Mel Spectrogram. This

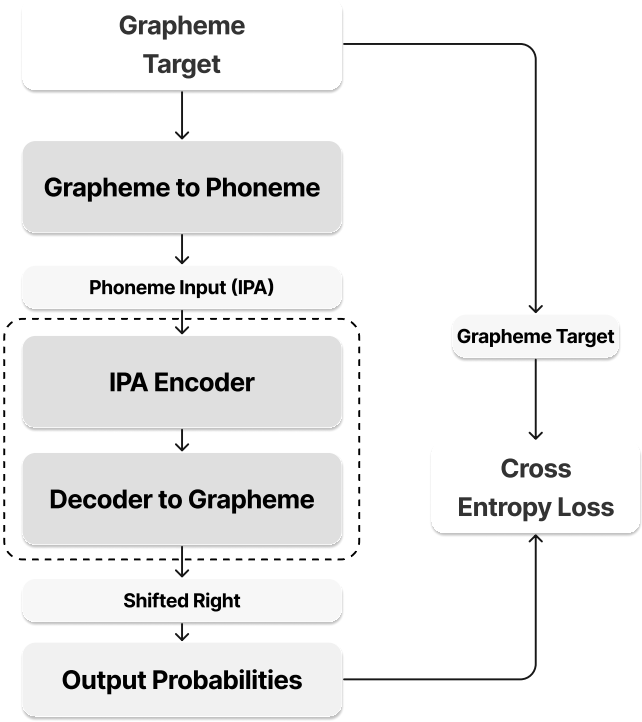


Fig. 4.8 The training architecture of Sound-skwatter without acoustic feedback from Mel Spectrogram reconstruction. This model functions as a baseline for assessing the impact of reconstruction on overall performance.

feature reduces training complexity and time and it was used in other Speech to Text proposals such as FastSpeech [49].

4.3 `Sound-squatter`: Multi-language Sound-squatting Generation

This section outlines the exploration of the sound-squatting generator, focusing on adapting the model to handle multi-language generation. Much of its content is derived from the paper titled *Lost in Translation: AI-based Generator of Cross-Language Sound-squatting*, presented at the 2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). The model builds upon the methodology applied in Section 4.2, adapting it to a multilanguage generation scenario.

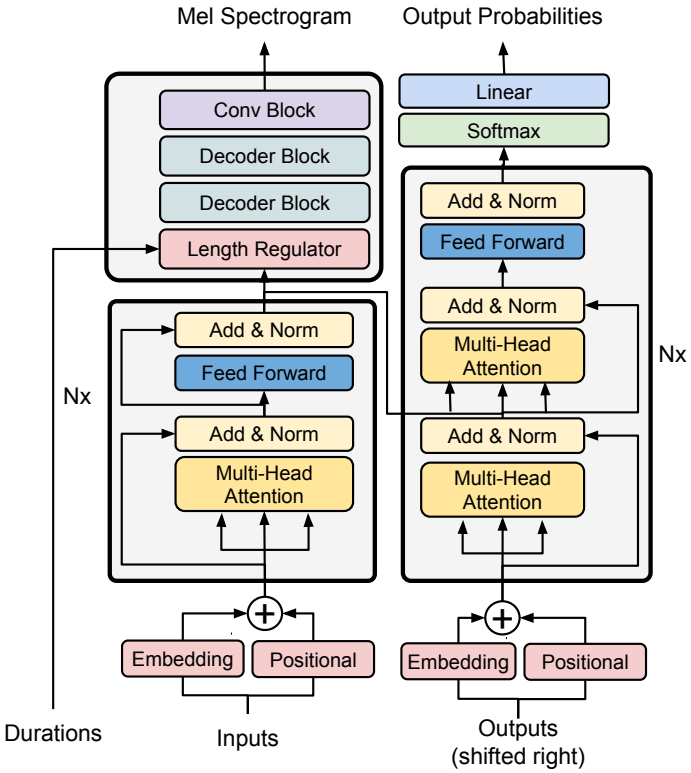
4.3.1 System Description

`Sound-squatter` builds on top of `Sound-skwatter`, therefore, its architecture (shown in Figure 4.10) contains the same components that are also present in `Sound-skwatter`. However, these components have been adapted to accommodate multiple languages within a single model. The generation pipeline comprises four primary components:

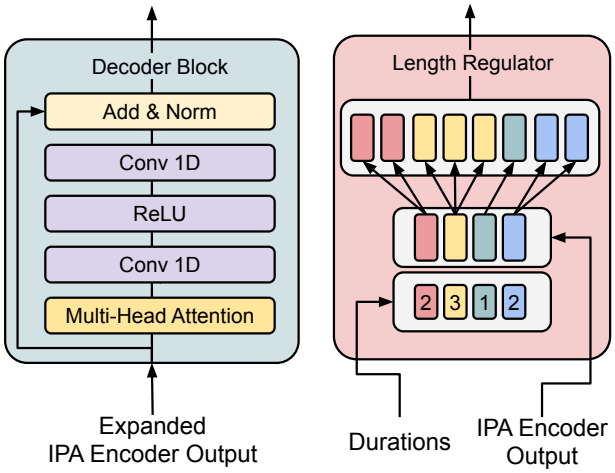
1. The ***Grapheme to Phoneme*** (G2P) component transforms an input word written in grapheme form (such as the word `eye`) into its corresponding International Phonetic Alphabet (IPA) representation (such as `/aɪ/` for British English).

2. The ***IPA Encoder*** component encodes the IPA word into a vector latent representation.

3. The ***Grapheme Decoder*** (P2G) component interactively decodes the vector representation into characters to compose graphemes form that are quasi-homophones of the input word (such as "`I`") in a target language.



(a)



(b) (c)

Fig. 4.9 (a) Full architecture of training; (b) Inside view of the Decoder Block; (c) High-level representation of the Length Regulator.

4. The ***Post Processor*** component performs beam search over the logits and selects tokens based on the probability from the decoder's output, resulting in the generation of multiple quasi-homophones from a single pronunciation.

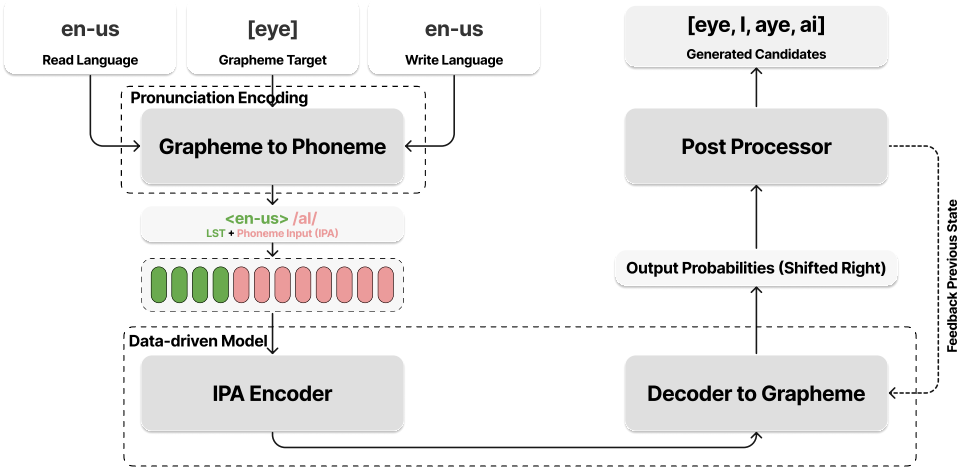


Fig. 4.10 Architecture used during inference. The process to generate candidates comprises an *IPA Encoder* that maps the input to a latent space and a *Grapheme Decoder* that produce several alternatives to reconstruct the input.

Grapheme to Phoneme

The usage of eSpeak NG (Next Generation) text-to-speech engine[23] is maintained, but now, in a multi-language setting the usage of Epitran[41] becomes necessary for transliterating text into IPA. The option of adding another tool is explained by the fact that eSpeak NG is a better fit English-GB and English-US, while Epitran is more suitable for other languages because eSpeak NG often switch languages when transliterating known words, which breaks the core idea of this proposal.

IPA Encoder and *Grapheme Decoder*

`Sound-skwatter` trainable components remain *IPA Encoder* and *Grapheme Decoder*. The state-of-the-art Transformer Neural Networks continue to be used. The

tool is designed as a single multi-language model with the explicit capability of generating cross-language homophones and quasi-homophones.

The control over the language used to read the grapheme used the language or accent of the G2P model. For example, the word "water" has different pronunciations in English-US (/ˈwɑtəɹ/), English-GB (/ˈwɔːtə/).

To specify the language the transformer should use to transliterate the phoneme back into grapheme form, language tokens are added to the input the "Language Special Token" (LST). These tokens provide contextualized information in the input conditioning the latent representation. In a nutshell, during training, the LST provides the input and output language information to the transformer, which learns to transliterate the phoneme based on the information contained between them.

The *Grapheme Decoder* (P2G) component is architecturally unchanged. This module is a Transformer Decoder module that interactively decodes the vector representation into characters to compose graphemes form that are quasi-homophones of the input word in a target language.

Post Processor

The *Post Processor* is updated. It continue receiving as input the *Grapheme Decoder*'s probabilistic forecasts of the next character, and it keeps track of the history of predictions to feedback to *Grapheme Decoder*. However, the usage of Beam Search changes from `top-k` to `top-p`. In the `top-k` approach, fixing the number of next tokens to select results in the selection of tokens with very low probabilities, as depicted in Figures 4.6, where the first branch occurs with one child being highly likely (b with probability of 0.9972) and another one being very unlikely to be the first character (p with probability of 0.0005) for the input pronunciation. This characteristic of `top-k` leads to the generation of many "bad candidates", which is avoided in the `top-p` approach.

In the `top-p`, the *Post Processor* picks from among the tokens those whose probabilities add up to p. Figure 4.11 shows the exact output for four iterations. At each step, the *Post Processor* stores C most-likely predictions of *Grapheme Decoder* whose probabilities add up to at least p and constructs alternative histories for the next step. Figure 4.11 shows this process with a directed graph diagram (with $p = 0.8$) starting from the IPA representation of `eye`. After four iterations,

the process generates six ways to write eye. Each branch stops when *Grapheme Decoder* outputs the special character EoS.

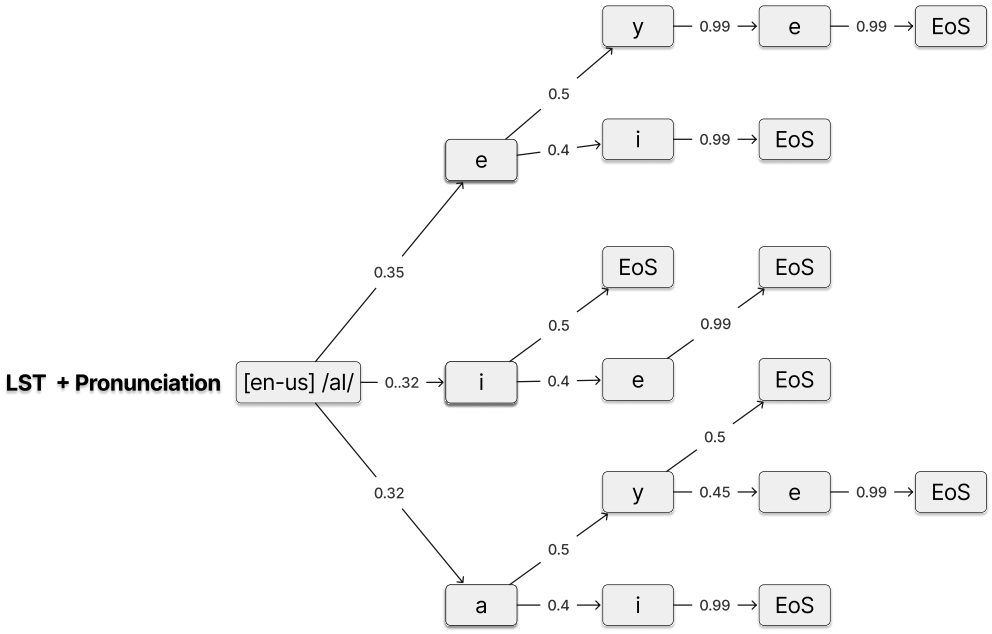


Fig. 4.11 Illustration of the inference process with $p = 0.8$. At each inference step, we explore n next characters whose probabilities add up to p. For readability, we round probabilities in the figure.

The number of iterations (M), maximum number of candidates predictions (K) and probability (p) are parameters that we can define manually. We again empirically define the parameters as follows: *Post Processor* iterates $M = N + 6$ times, where N is the size of the source string, and we limit exploration with $K = 100$ and $p = 0.8$, where K is the maximum number of possible candidates generated.

4.3.2 Dataset and Training

Four different languages were selected for training: English-US, English-GB, French-FR, and Portuguese-BR. English was chosen because it is the most widely used language on the Internet, and including two English variations is important for illustrating factors related to homophone-based impersonation. French-FR was also chosen because it is not a phonetic language, making it more prone to confusion

Table 4.3 Sound-squatter Dataset size for each chosen language.

Language Tag	Language	Region	Data size
en-GB	English	United Kingdom	65 118
en-US	English	United States	125 923
fr-FR	French	France	245 971
pt-BR	Portuguese	Brazil	95 943
	Total		532 955

during transliteration. Portuguese-BR is more regular compared to all the languages before, being a Phonetic language.

Phonetic languages exhibit a close correspondence between pronunciation and written representation, with each letter or character representing a specific sound. While French and English have some phonetic elements, they are less purely phonetic compared to languages with more straightforward sound-to-spelling correspondences, such as Portuguese, Italian and Spanish.

The training dataset comprises the list of English-US, English-GB, Portuguese-BR and French-FR words from the GNU Aspell [7] word list. GNU Aspell is a free and open-source spell checker containing word lists for multiple languages. To acquire the pronunciation, rule-based G2P tools are used: eSpeak NG for English-GB and English-US, and Epitran for French-FR and Portuguese-BR.

Table 4.3 displays the size of the dataset for each language. In total, 532 955 words and their pronunciations are used.

The *IPA Encoder* and the *Grapheme Decoder* are trained with a batch size of 64 words. The training set contains 80% of the samples, while 10% are preserved for validation and 10% for test sets. The maximum sequence length is 50 tokens. The validation set is used for selecting the best model, and the test set is used for verifying overfitting. The Adam optimizer is employed with $LR = 0.0001$, $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\varepsilon = 10^{-9}$. The model is trained for 10 epochs, taking around 10 minutes on a single NVIDIA Tesla v100. The Transformer Network hyperparameters for the encoder and the decoder are symmetric and defined as follows: the size of the hidden representation is 512, the embedding dimension is 512, the number of heads is 8, and the vocabulary size is 123.

Table 4.4 `Sound-squatter` Hyperparameters

Hyperparameter	Value
Number of heads	8
Latent dimension	512
Maximum Sequence Length	50 tokens
Epochs	10
Batch Size	64
Training dataset split	80%
Validation dataset split	10%
Testing dataset split	10%
Optimizer	Adam
Learning Rate	0.0001
ε	10^{-9}
Training Epochs (Model)	10
Training Time (Model)	$\approx$ 10 minutes
Hidden Representation Size	512
Embedding Dimension	512
Vocabulary Size	123

4.3.3 Architecture Details

Figure 4.12 provides a detailed overview of the model architecture used for homophone generation. The diagram illustrates the components, including the *IPA Encoder*, *Grapheme Decoder*, and the Language Token Embedding module. The Language Token Embedding module receives the LST and converts it to the same dimensionality as the embedded IPA tokens before forwarding it to the *IPA Encoder*.

4.4 `X-Squatter`: Cross-language Sound-squatting Generation

This section introduces `X-Squatter`, which stands out from previously presented tools as a hybrid model. This hybrid model uses Articulatory Feature Vectors to establish the connection between IPA tokens and audio signals. Most of the section content is derived from the paper titled *X-squatter: AI Multilingual Generation of Cross-Language Sound-squatting*, published at the ACM Transactions on Privacy and Security.

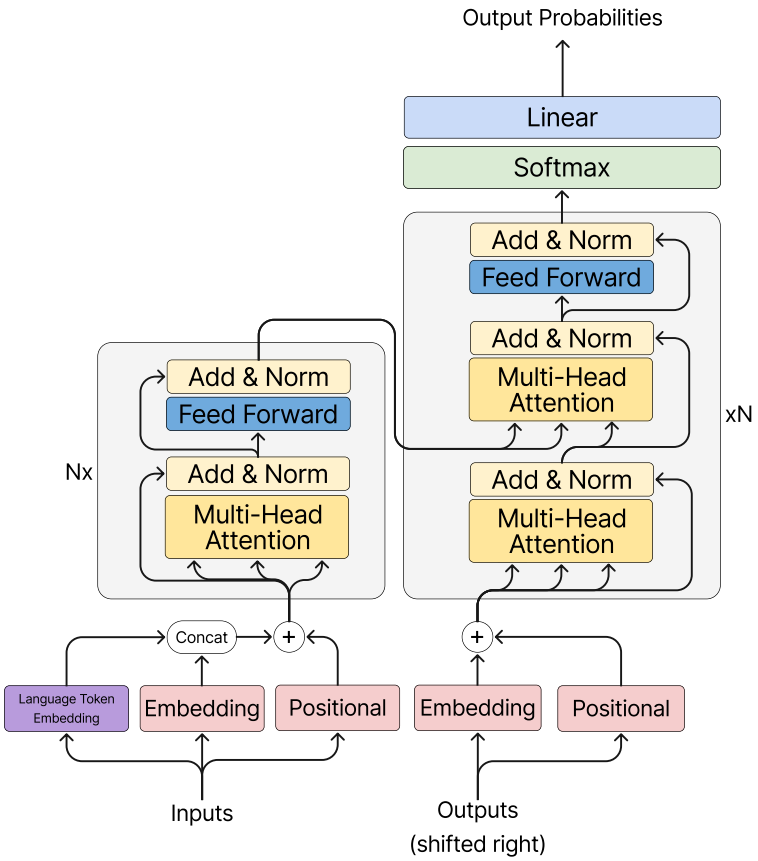


Fig. 4.12 Complete training architecture of Sound-squatter, featuring a Transformer Neural Network augmented with a language token concatenation module.

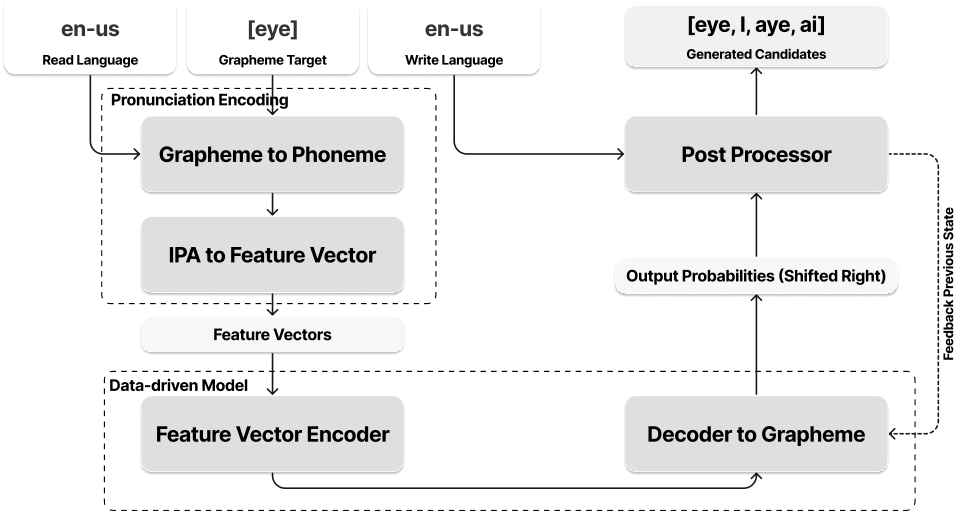


Fig. 4.13 Architecture used during inference. The process to generate candidates comprises an *Feature Vector Encoder* that maps the input to a latent space and a *Grapheme Decoder* that reconstructs the input.

4.4.1 System Description

`X-Squatter` (pronounced `cross-squatter`) is a tool crafted for the automatic generation of single- and cross-language sound-squatting candidates. It leverages Articulatory Feature Vectors to model any phonemic representation, enabling the substitution of phoneme tokens with counterparts that share similar auditory characteristics.

Operating at the sub-word level, `X-Squatter` generates candidates from a target name while incorporating mechanisms to control candidate quality during the search process, akin to its predecessors. It takes inputs of the word's written form (*grapheme*), its pronunciation represented in the International Phonetic Alphabet (IPA), and the language of origin. By extracting features from pronunciation segments, it produces homophones seamlessly across multiple languages and clear the way for cross-language scenarios, necessitating innovative approaches to handle graphemes for non-existing phonemes in the target language.

`X-Squatter` improves the translation process of phonemes into corresponding graphemes by leveraging Articulatory Feature Vectors. The workflow is depicted

in Figure 4.13, which shows a generation pipeline built upon five key components, some of them which are in common with the previous proposals:

1. The ***Grapheme to Phoneme*** (G2P) component converts the input word presented in grapheme format (in the example, the word "eye") into its corresponding representation in the International Phonetic Alphabet (IPA), given a read language (such as /aɪ/ for British English).

2. The ***IPA to Feature Vector*** component processes each segment of the IPA representation, substituting it with a feature vector that aligns with acoustic attributes.

3. The ***Feature Vector Encoder*** component transforms the sequence of feature vectors into representation in a latent space using a Transformer-based sequence-to-sequence model.

4. The ***Grapheme Decoder*** (P2G) component interactively decodes the latent representation into characters to compose a new grapheme form that presents similar pronunciation of the input word (such as "I"). The initial state fed to the model selects the target language.

5. The ***Post Processor*** component engages in beam search across the output logits and makes token selections according to probabilities derived from the decoder's output.

The model receives three inputs the (grapheme target, read language, write language), where the "grapheme target" is the word for the creation of sound-squatting candidates is desired. The "read language" represents the language used to pronounce the word, as the same sequence of letters can have different pronunciations in different languages. Similarly, the "write language" denotes the language in which the transcription of the pronunciation is done (i.e., the language of the sound-squatting victim). The output of the tool is a set of sound-squatting candidates that belong to the target language and include homophones, quasi-homophones, and words with similar pronunciation.

Grapheme to Phoneme (G2P)

`X-Squatter` continues on using eSpeak NG engine and Epitran for transliterating text into IPA.

IPA to Feature Vector

A novel addition to the pipeline involves the role of the *IPA to Feature Vector* module. IPA segments are mapped to respective *Articulatory Feature Vectors* by this module. This technique produces a rich representation of IPA segments, introducing a concept of similarity that is absent in pure IPA representation. Since IPA is a symbolic alphabet, it is not possible to measure how similar two IPA segments are, such as in terms of the sound of their pronunciations. This understanding of similarity becomes crucial for cross-language sound-squatting generation, as each language may include only a subset of IPA segments. When dealing with two different languages, it becomes necessary to search within the destination language set for other phonemes that can effectively replace those from the original language. Additionally, the definition of a metric that quantifies the similarity between phonemes allows for the search for quasi-homophones and words with similar pronunciations, better controlling the quality of the generated candidates. In this application, the similarity metric is implicit and guided by the transcription results.

Projects like [40, 39, 22] aggregate IPA segments and their features for various languages. Implementations like Uriel, Phonological Mapping [33, 55], and PanPhon [42] convert IPA segments into articulatory feature vectors for NLP tasks. PanPhon, for instance, maps over 6 000 IPA segments to 21 subsegmental articulatory features and is used in `X-Squatter`.

Feature Vector Encoder and *Grapheme Decoder*

With the introduction of *IPA to Feature Vector*, the adaptation of the *Feature Vector Encoder* becomes necessary. The Embedding block in the IPA Encoder (See Section 4.4.3) is replaced by a Feature Vector Encoder. This change is necessary because the Embedding block converts tokens to vectors, meaning that the input is a sequence of IPA tokens. Now, the *Feature Vector Encoder* receives a sequence of feature vectors as input, which have different dimensions.

On the decoder side, the model takes the latent representation build from the feature vectors and generate (i) the corresponding ISO code of the language, (ii) the word's written pronunciation form. For example, given the feature vector of the input "/ˈwɑtəɹ/", the correct output sequence is "en-us water". Similarly, if provided with "/ˈwɔːtə/", the anticipated output should be "en-gb water". When faced with such tasks, the model associates certain combinations of IPA segments to specific languages or accents.

The design of X-Squatter as a multi-language model with the explicit capability of generating cross-language homophones and quasi-homophones requires the control of the language used to read the grapheme by changing the language or accent of the G2P model. In the cross-language case, for example, suppose the pronunciation of "water" in English-US and specify that we want the grapheme form to be transliterated into French-FR. In that case, the model can generate the quasi-homophone "warères" (/waʀəʀ/), which does not exist as a word in French-FR but has a similar pronunciation to "water" in English-US.

To specify the language the transformer shall use to transliterate the phoneme back into grapheme form, in the start of the decoder input the target language ISO code.

Post Processor

The same Beam Search top-p strategy is used by the *Post Processor*. Parameters such as the number of iterations (M), maximum number of candidate predictions (K), probability (p), and temperature (t) can be manually defined.

However, at the beginning of grapheme generation, a seed is now used to condition the generation. This seed corresponds to the "write language" as used in training. This strategy is somewhat common in GPT models, where the output of the model is a continuation of the previous tokens. However, X-Squatter is not a GPT model because it consists of both an encoder and a decoder (GPT models are decoder only), although the reasoning behind the strategy remains the same.

Additionally, compared to Sound-squatter, where LST are concatenated, the architecture in this X-Squatter is cleaner and more homogeneous.

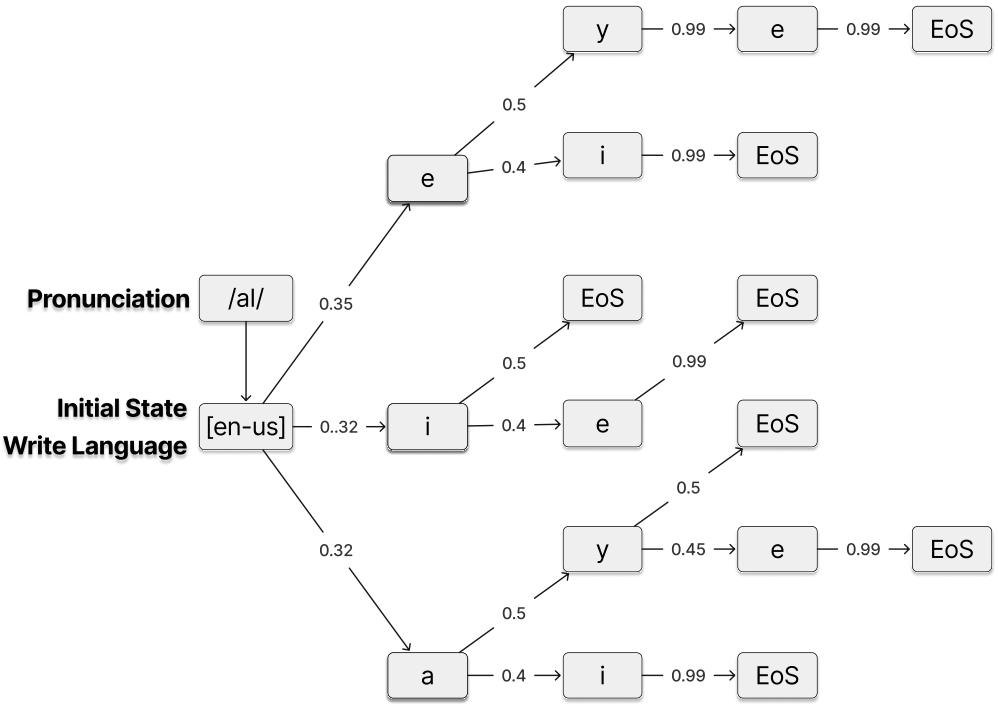


Fig. 4.14 Illustration of the inference process with $p = 0.8$. At each inference step, we explore n next characters whose probabilities add up to p. For readability, we round probabilities in the figure.

Table 4.5 X-Squatter Dataset size for each chosen language.

Language Tag	Language	Region	Data size
en-GB	English	United Kingdom	65 118
en-US	English	United States	125 923
fr-FR	French	France	245 971
pt-BR	Portuguese	Brazil	95 943
	Total		532 955

4.4.2 Dataset and Training

This section outlines the training procedure for X-Squatter. The model undergoes training using four distinct languages: English-US, English-GB, French-FR, and Portuguese-BR. Restricting the training to proto-Indo-European languages still accounts for approximately 66.8% of online web content [47].

The training dataset comprises English-US, English-GB, French-FR, and Portuguese-BR words sourced again from the GNU Aspell [7] word list. Pronunciations are obtained using rule-based G2P tools, namely eSpeak NG for English-GB and English-US, and Epitran for French-FR and Portuguese-BR.

Table 4.5 illustrates the dataset sizes for each language, totaling $437,012$ words and their respective pronunciations. Both *Feature Vector Encoder* and *Grapheme Decoder* are trained with a batch size of 16 words, where 80% of the samples are used for training, 10% for validation, and another 10% for testing.

The training employs the Adam optimizer with a learning rate (*LR*) set at 0.0001, along with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\varepsilon = 10^{-9}$. Training halts upon observing three consecutive epochs of no decrease in Validation Loss. Hyperparameters for the Transformer Network include a hidden representation size of 512, an embedding dimension of 512, and 8 attention heads, symmetrically applied to both encoder and decoder modules. The input vocabulary includes $6,487$ IPA segments, while the output vocabulary consists of 133 grapheme tokens, standardized across all languages. Table 4.6 enumerates these hyperparameters.

During training, an IPA input is paired with the associated language's ISO code and the word in its written form. Figure 4.15 outlines the training process of X-Squatter, with dashed lines demarcating the boundaries of learnable parameters.

Notable distinctions in `X-Squatter`'s training process compared to previous models is the direct passage of the "write language" to the *Grapheme Decoder* block without encoding in latent features.

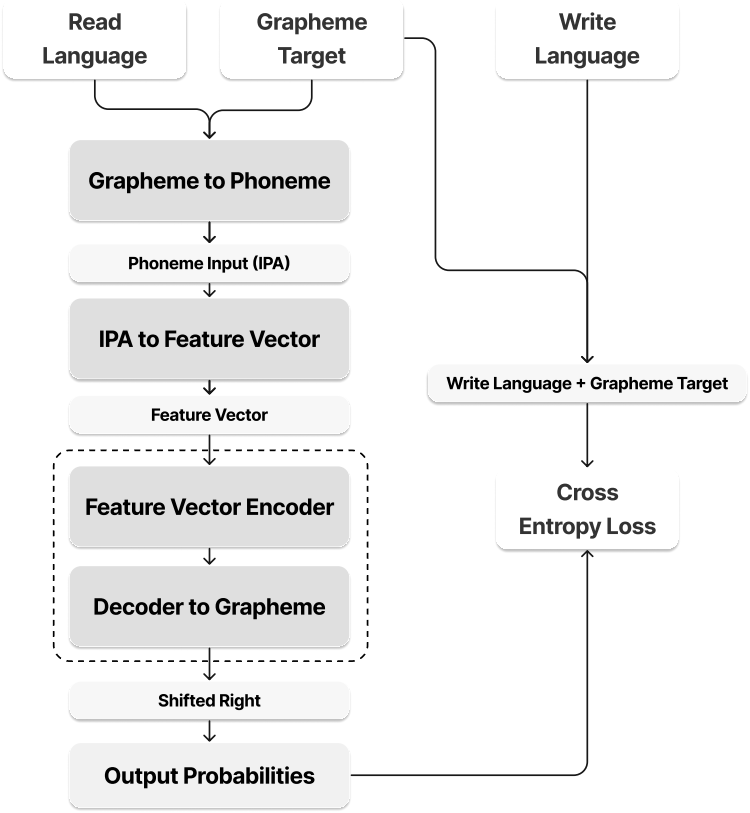
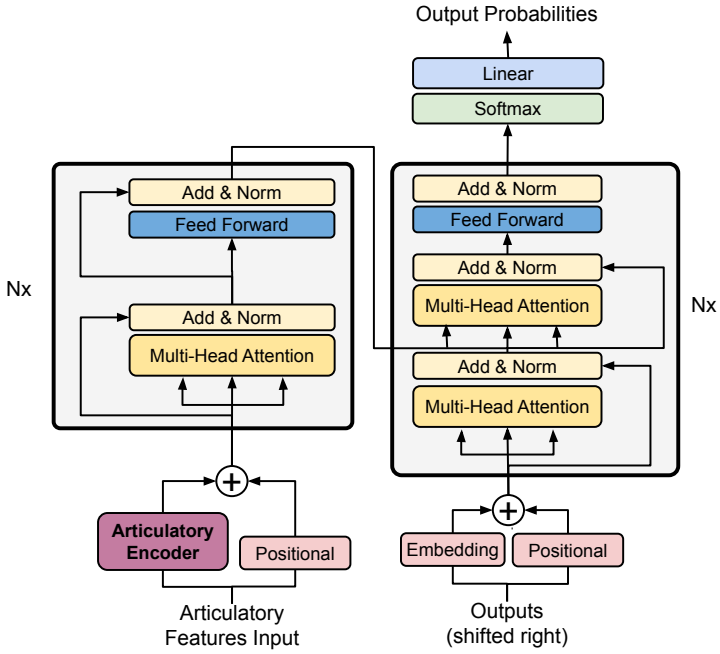


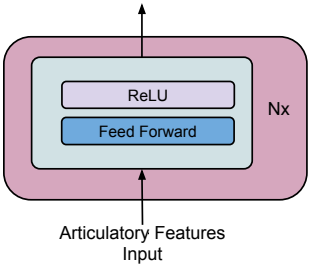
Fig. 4.15 Training process of `X-Squatter`, with dashed lines denoting trainable modules. Third-party modules, including *IPA to Feature Vector* and *Grapheme to Phoneme*, are integrated into the current pipeline.

4.4.3 Architectural Details

Figure 4.16a details the architecture of the model used for homophone generation. It includes the internal of the *Feature Vector Encoder* and *Grapheme Decoder*. Figure 4.16b shows the inside of the decoder block that is used to reduce the complexity of the visualization.



(a)



(b)

Fig. 4.16 (a) High-level representation of the X-Squatter Transformer architecture. (b) Inside view of the Articulatory Decoder Block.

Table 4.6 `X-Squatter` Hyperparameters

Hyperparameter	**Value**
Number of heads	8
Latent dimension	512
Maximum Sequence Length	50 tokens
Epochs	38
Batch Size	16 words
Training dataset split	80%
Validation dataset split	10%
Testing dataset split	10%
Optimizer	Adam
Learning Rate	0.0001
ε	10^{-9}
Training Steps (Model)	1157849
Validation Loss Stopping Criteria	3 consecutive epochs
Hidden Representation Size	512
Embedding Dimension	512
Input Vocabulary Size	6 487 IPA segments
Output Vocabulary Size	133 grapheme tokens

Chapter 5

Results and Validation

In this chapter, we present the results of the study's validation, organized into two main sections: quantitative evaluations related to the tools (Section 5.1) and a qualitative evaluation concerning the anticipation of user's transcription mistakes (Section 5.2).

5.1 Tool's Validation

The validation of the data-driven alternatives we propose for generating homophones focus on their capability in producing well-accepted written forms for specific pronunciations. This validation is based on assessing the coverage of known homophones given a target word. The higher the coverage of known homophones by the tool when generating candidates, the better its generation capacity. Coverage is evaluated for exact homophones within the same language (Section 5.1.1) and across different languages (Section 5.1.2). Another aspect of validation involves quasi-homophone generation. The validation of the tools regarding the quasi-homophone generation involves assessing the similarity in terms of Articulatory Feature Edit Distance. The tools are validated for quasi-homophones within the same language (Section 5.1.3) and across different languages (Section 5.1.4).

5.1.1 Single-language Homophone Coverage

To evaluate single-language homophone coverage, we use a curated list of known homophones provided by the AIL tool (refer to Section 2.4) in English-US. Table 5.1 presents some examples of homophones extracted from this list. The list comprises 362 pronunciations in English-US, each represented as IPA transcripts. Each IPA transcript corresponds to at least two English words, rendering them homophones. This curated list serves as the ground truth dataset. The evaluation process consists in inputting the IPA transcripts into all tools and gathering the candidates generated by each model. When predictions are made using the models, the beam search is configured with a probability threshold of $p = 0.9999$ and a temperature value of $t = 1.0$.

Table 5.1 Examples of homophones obtained from their IPA representations.

IPA Pronunciation	Homophones
waɪt	white, wight
slaɪt	sleight, slight
ɜːn	earn, urn
bɔl	ball, bawl
neɪ	nay, neigh

We measure the ratio of known homophones generated by each model for every pronunciation. Exact homophones can have a variety of written forms (refer to Table 5.1). If a model produces x known forms for a specific pronunciation with y known written forms, the ratio x/y is calculated. The performance of each tool for every K is assessed by the average coverage for all n pronunciations: $\frac{1}{n}\sum_{i=1}^{n}\frac{x_i}{y_i}$ where $n = 362$ pronunciations.

Results are summarized in Figure 5.1, depicting the average coverage with lines, while the color ranges denote the 95% confidence interval. The numbers for $K = 35$ are presented in Table 5.2. In terms of the coverage metric, several findings emerge. First, `Auto-Squatter` performs poorly. An average below 0.5 suggests that `Auto-Squatter` sometimes struggles to even identify a single correct written form for the pronunciation, given that every group of homophones has at least two written forms. Second, the incorporation of a base architecture along with the beam search significantly impacts coverage. `Sound-skwatter`, with and without audio feedback, and `Sound-squatter` exhibit identical averages and standard deviations, thus indicating that the usage of audio feedback has no impact on generation.

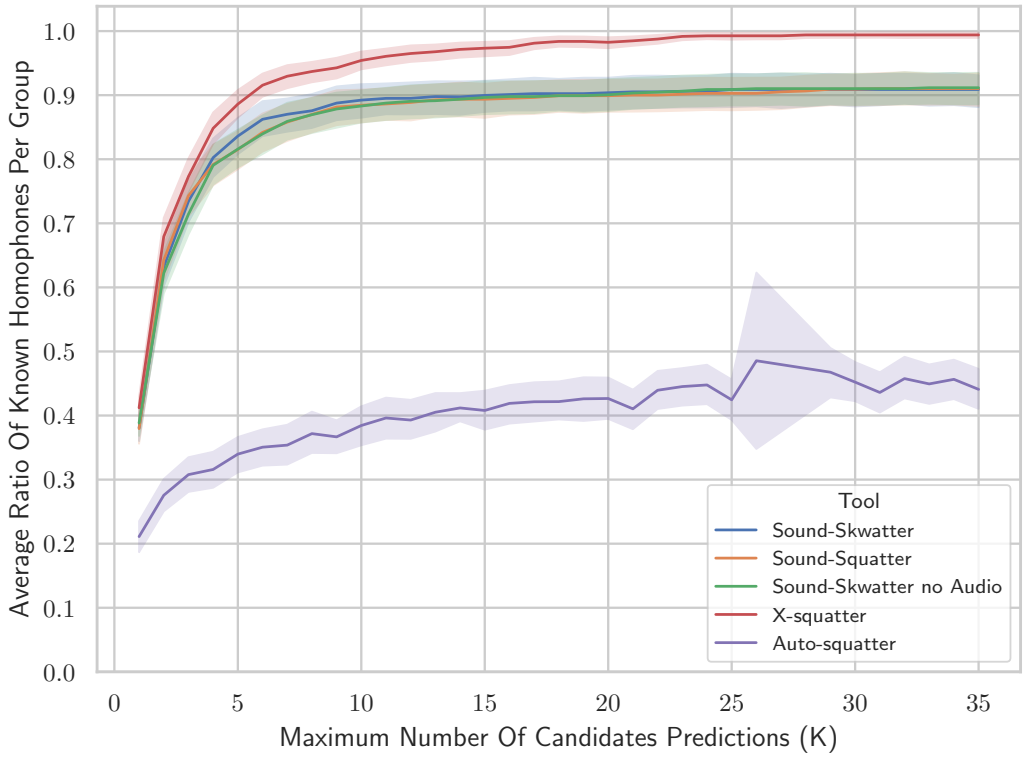


Fig. 5.1 Homophone coverage for single-language scenario. As the maximum number of generated candidates increases (*Post Processor K* parameter), the model exhibits a higher coverage. The 95% confidence internal is shown.

Table 5.3 displays the absolute number of missing homophones for $K = 35$. `X-Squatter` produces all candidates, while the other models perform similarly. The exception once again is `Auto-Squatter`, which misses 434 written forms out of 766 in the evaluation set.

Table 5.2 Performance metrics for each tool at $K = 35$ in the single-language scenario, where "Average Coverage" indicates the mean proportion of known homophones generated per group, and "Standard Deviation" denotes the variability of coverage across the evaluation set.

Tool	**Average Coverage**	**Standard Deviation**
`Auto-Squatter`	0.44	0.31
`Sound-skwatter`	0.91	0.24
`Sound-skwatter` no Audio	0.92	0.24
`Sound-squatter`	0.92	0.24
`X-Squatter`	1.00	0.00

Table 5.3 The absolute number of missing homophones for each model at $K = 35$, indicating the count of homophones that are not generated by the respective tool.

Tool	**Missing Homophones**
`Auto-Squatter`	434
`Sound-skwatter`	67
`Sound-skwatter` no Audio	65
`Sound-squatter`	66
`X-Squatter`	0

5.1.2 Cross-language Homophone Coverage

This validation verifies the coverage each tool performs in known homophones in cross-language scenario. All pronunciations are collected from the `X-Squatter` and `Sound-skwatter` training dataset, and words with exactly the same pronunciation across different languages are grouped together. It is important to note that the models are trained using phoneme/grapheme pairs of the same language and have not encountered cross-language homophones during training.

Due to the strict requirement that the pronunciation must be identical across at least two languages, the number of cross-language homophones is relatively small compared to the entire dataset. A total of 95 pronunciations with multiple written forms in various languages have been collected. In total, it is observed 374 written forms spanning four languages. Examples of cross-language homophones are shown in Table 5.4.

Table 5.4 Examples of exact cross-language homophones with IPA Pronunciations.

IPA Phoneme	Homophones
ɛʃ	$[fr-fr]$ ais, $[en-us]$ esse
ʃɪ	$[fr-fr]$ chie, $[pt-br]$ xi, $[en-us]$ shi
kædi	$[fr-fr]$ caddie, $[pt-br]$ cádi, $[en-gb]$ caddie, $[en-gb]$ caddy
maʃ	$[fr-fr]$ mâchai, $[pt-br]$ más, $[en-gb]$ mache, $[en-gb]$ mash
kaʃ	$[fr-fr]$ cache, $[pt-br]$ chás, $[en-gb]$ kasch

Following the same approach as for the single language validation, we measure the ratio of known homophones generated by each model to each group of homophones.

A grid search is performed, varying the read language, write language for each target grapheme. During predictions, a probability threshold $p = 0.9999$ and a temperature value $t = 1.0$ are used again.

Figure 5.2 provides a visualization of the results. Similar to the single-language scenario, increasing the maximum number of candidates systematically enhances the average coverage. It is evident that, just like in the single language scenario, `Auto-Squatter` performs poorly with a maximum average ratio of 0.3, meaning that it often fails to generate more than one known written form for the pronunciation.

It is also observed that the single language tool `Sound-skwatter` (both with and without audio feedback) perform poorly, as expected, since cross-language knowledge is required for the task. The best performance is achieved by the multi-language models `Sound-squatter` and `X-Squatter`, with `X-Squatter` having a slightly higher average of 0.95 (std 0.13), while `Sound-squatter` shows 0.92 (std 0.24) (See Table 5.5). This difference means that `Sound-squatter` misses 36 homophones while `X-Squatter` misses 25 out of 372, as shown in Table 5.6.

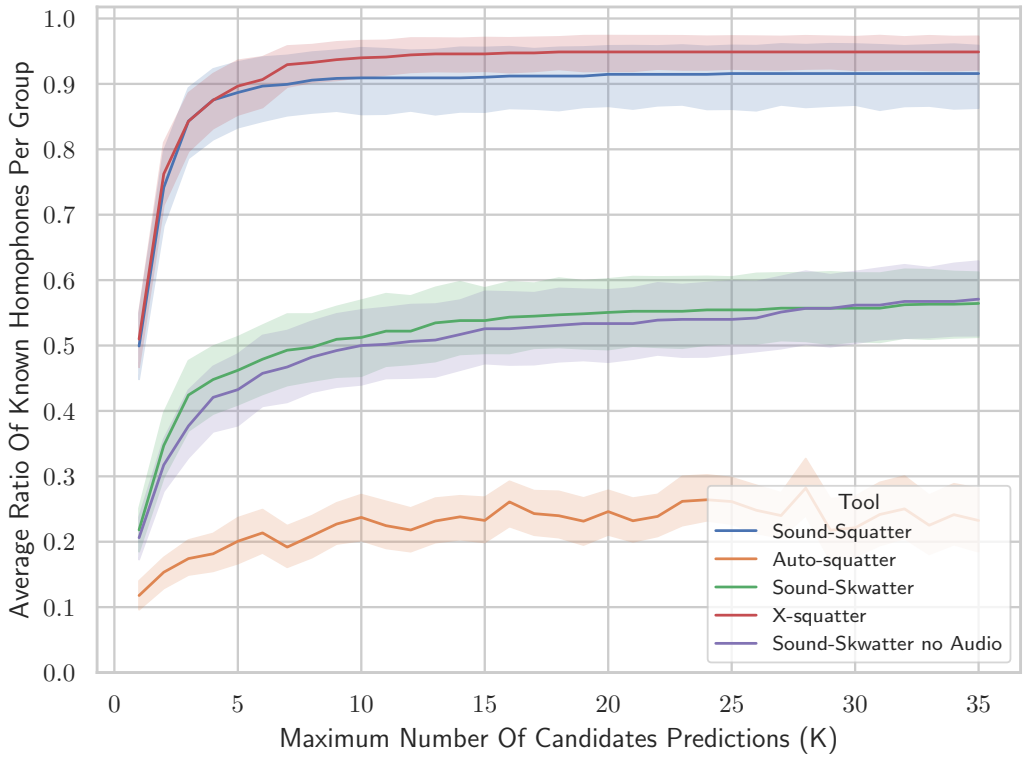


Fig. 5.2 Homophone coverage for cross-language scenario. As the maximum number of generated candidates increases (*Post Processor K* parameter), the model exhibits a higher coverage. The 95% confidence internal is shown.

Table 5.5 Performance metrics for each tool at $K = 35$ in the cross-language scenario, where "Average Coverage" indicates the mean proportion of known homophones generated per group, and "Standard Deviation" denotes the variability of coverage across the evaluation set.

Tool	**Average Coverage**	**Standard Deviation**
`Auto-Squatter`	0.23	0.25
`Sound-skwatter`	0.56	0.27
`Sound-skwatter` no Audio	0.57	0.28
`Sound-squatter`	0.92	0.24
`X-Squatter`	0.95	0.13

Table 5.6 The absolute number of missing homophones for each model at $K = 35$, indicating the count of homophones that were not generated by the respective tool.

Tool	**Missing Homophones**
`Auto-Squatter`	293
`Sound-skwatter`	175
`Sound-skwatter` no Audio	174
`Sound-squatter`	36
`X-Squatter`	25

5.1.3 Single-language Quasi-homophone Generation

The coverage analysis initially focused on exact homophones compiled by gathering exact matches in IPA encoding. However, to evaluate quasi-homophone generation, this analysis requires adaptation, since there is no list of quasi-homophones to measure coverage.

To systematically evaluate quasi-homophone generation, a set of homophones is generated using each tool, with parameters set to $K = 10$, $p = 0.9999$, and $t = 1.0$. The parameter K is selected balancing quality and coverage measured in previous sections. In the `Auto-Squatter` case, we set up 10 runs using the same reasoning. Subsequently, to assess the quality of the quasi-homophones generated, we compute a distance metric between the target word and each candidate. We randomly select 30 words, of which we use the pronunciation, from the homophone dataset used Section 5.1.1.

The distance metric is the "Weighted Feature Edit Distance", which represents a technique for measuring the dissimilarity between two feature vectors. This method incorporates the notion that the expenses linked with modifying features vary depend-

ing on their class and subjective variability [42]. Essentially, it acknowledges that certain features hold different levels of importance or relevance in distinct contexts or domains. Consequently, the cost associated with editing these features is adjusted accordingly. This approach enables a more nuanced and contextually sensitive evaluation of the similarity or dissimilarity between feature vectors, accommodating the diverse nature of real-world data. A function for computing this specialized edit distance for Articulatory Feature Vectors is provided by the panphon [42].

Figure 5.3 shows the distances measure for every pair of target and generated homophone generated by each tool, which gives a total of 1 754 pairs. In this case, it is restricted only to English-US language. The results for the distances are summarized in Figure 5.3 and detailed in Table 5.7.

Note that the only model that presents some deviation in the metric is `Auto-Squatter` with mean distance 4.65 ± 5.06. All models generate pairs which are on average very similar in terms of pronunciation. For comparison, the words *think* and *sink* have a Weighted Feature Edit Distance of 7.25, the pair *grass* and *glass* 1.5 and *travel* and *trouble* 8.375.

Comparing the other models, it is notable that the average distance is very close for all of them. However, there is a significant difference in the standard deviation. This difference in the standard deviation can be attributed to several factors.

`Sound-skwatter` is a single-language tool for English US, which matches the language used in this evaluation. Consequently, `Sound-skwatter` does not learn alternative mappings for pronunciation-spellings found in other languages, leading to homophones with fewer spelling variations.

In contrast, both `Sound-squatter` and `X-Squatter` are trained for multiple languages. The difference in their standard deviation is not due to the dataset, as they are both trained with the same data. The key distinction likely comes from `Sound-squatter` being token-based, while `X-Squatter` is a hybrid model. Hybrid models better account for the impact of spelling on pronunciation, allowing for greater flexibility in generating candidates and achieving higher pronunciation accuracy.

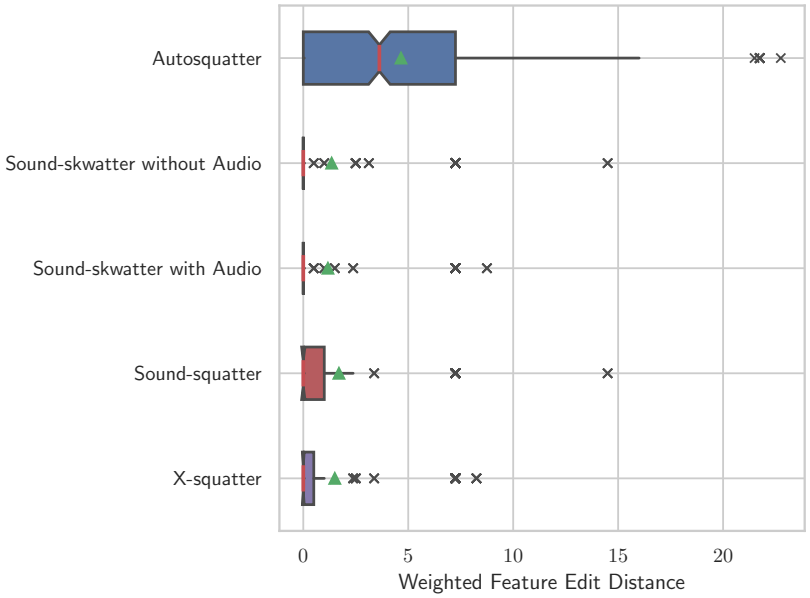


Fig. 5.3 Weighted Feature Edit Distance measured for every pair of target and generated homophone generated by each tool. The 1754 pairs are considered. The median value is represented by the red line and the outliers marked by the "x" and mean is indicated by a triangle.

Table 5.7 Performance metrics for each tool, where "Average Distance" indicates the mean Weighted Feature Edit Distance, and "Standard Deviation" denotes the variability of distances across the evaluation set.

Tool	**Unique Quasi-homophones**	**Average Distance**	**Standard Deviation**
`Auto-Squatter`	492	4.65	5.06
`Sound-skwatter` with Audio	408	1.17	2.65
`Sound-skwatter` without Audio	381	1.35	2.89
`Sound-squatter`	401	1.70	3.12
`X-Squatter`	346	1.50	2.88

5.1.4 Cross-language Quasi-homophone Generation: Impact of *Feature Vector Encoder*

An issue on the quasi-homophone generation in the cross-language scenario happens when the set of IPA segments present in each language are not equal. For this reason, some similar pronunciations in two different languages might be impossible to discover because of the missing links in representation.

Consider as an example the word "gnocchi," pronounced as /ˈɲɔk.ki/ in Italian. Some IPA segments in the Italian pronunciation are not present in the en-US (American English) phonetic inventory. Consequently, the Italian pronunciation of "gnocchi" encompasses sounds that do not precisely align with American English phonology.

The *Simple IPA Translation* `Auto-Squatter`, `Sound-skwatter` no Audio, and the *Audio Inbound with IPA Translation* `Sound-skwatter` models cannot properly handle these cases, as they must be trained for a particular language to understand the phonetic inventory of that language. Therefore, to maintain fairness in the analysis, this examination is limited to `Sound-squatter` and `X-Squatter`. In essence, this analysis verifies the impact of the *Feature Vector Encoder* in the model.

To perform an empirical evaluation of how the two models deal with gaps in the representation, we perform two experiments, considering cases where there are (or there are not) gaps in the representation across languages.

We first select 30 target words. These words are presented to both tools and the candidate homophones are collected. we exclude the known homophones and thus the evaluation contains only quasi-homophones. The generation is configured with parameters $K = 10$, $p = 0.9999$, and $t = 1.0$. The read language is English-US and the Write Language French-FR (results are consistent in other setups). Therefore, the quasi-homophones are all, by definition, cross-language homophones. This group of quasi-homophone *without representation* gaps contains 3 144 pairs of targets and quasi-homophone.

In the second experiment, we collect a set of 60 000 Italian words from GNU dictionary. A subset of these Italian words are selected, in which the phonemic representation contains segments exclusive to Italian such as ɲ and ʎ, summing up to 20 segments. Using the same parameters $K = 10$, $p = 0.9999$, and $t = 1.0$, but Read Language Italian-IT and Write Language English-US, we produce 982 pairs

of targets and generated homophones for 30 randomly selected target words in this subset.

Figure 5.4 shows the Weighted Feature Edit Distance distribution metrics for all pairs separated by quasi-homophone set and tool. Table 5.8 shows that for `Sound-squatter`, the average distance is 13.80 with phonemic gaps and 7.62 without, while for `X-Squatter`, the average distances are 9.31 and 8.07 respectively. Standard deviations show greater variability with phonemic gaps, particularly evident in `X-Squatter` with a standard deviation of 9.95.

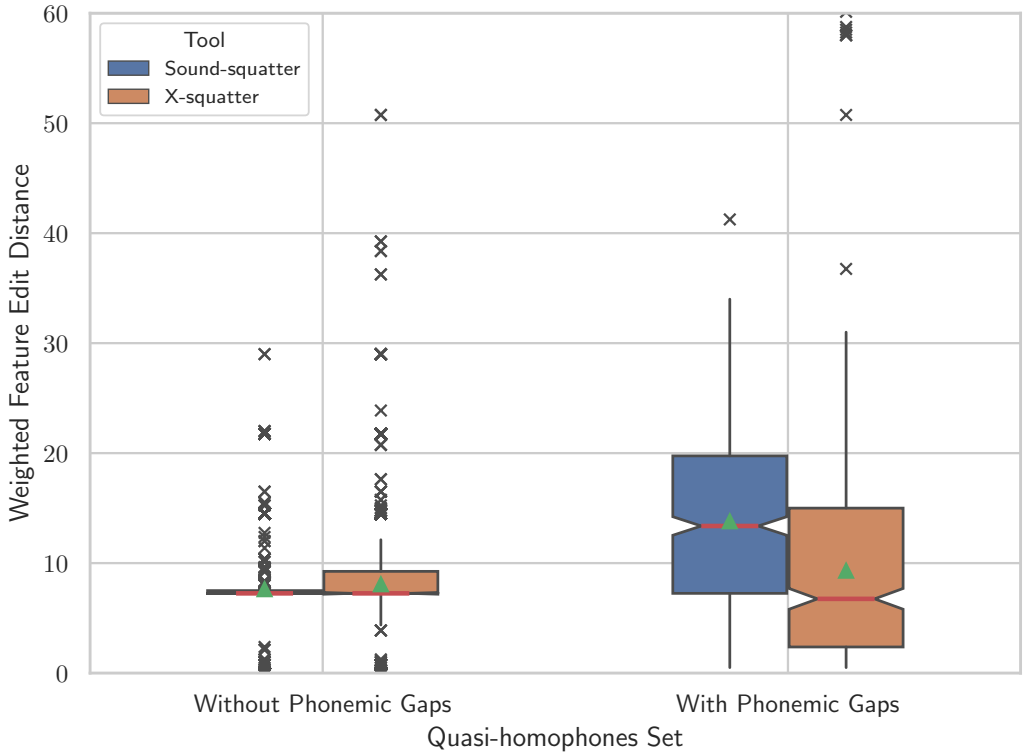


Fig. 5.4 Weighted feature edit distance calculated over pairs of input targets and homophone candidates generated by a specific tool. The metric captures the similarity in the pronunciation. The two sets of input words are input words that are exact homophones and the phonemic representation is seem during training in both models. The other set of input words are for target words in the Italian language. The targets are selected specifically because it contains IPA segments that are not seem during training. The median value is represented by the red line and the outliers marked by the "x" and mean is indicated by a triangle.

Since, `X-Squatter` uses a hybrid approach to produce candidates for a given target, it presents a distinct advantage in its capacity to find good substitutes for IPA tokens not present in the spelling language. This is possible by the incorporation

Table 5.8 Comparison of Quasi-Homophone Weighted Feature Edit Distance Across Tools and Phonemic Gap Conditions

Tool	Quasi-homophone Set	Quasi-homophone Count	Average Distance	Standard Deviation
`Sound-squatter`	With Phonemic Gaps	540	13.80	7.35
	Without Phonemic Gaps	1455	7.62	4.81
`X-Squatter`	With Phonemic Gaps	442	9.31	9.95
	Without Phonemic Gaps	1689	8.07	5.73

of Articulatory Feature Vectors within the model, which align tokens based on articulatory attributes – see the lower mean distance measure in its quasi-homophone candidates.

5.1.5 Key Insights from Coverage Validation

The validation of data-driven tools for generating homophones demonstrates their ability to produce accurate written forms based on specific pronunciations. For single-language homophone coverage, the tools were assessed using a list of 362 English-US homophones represented in IPA. The performance of each tool was evaluated based on the ratio of known homophones generated for each pronunciation. The results indicate that `Auto-Squatter` performed poorly, with an average coverage below 0.5, often failing to generate even a single correct written form. In contrast, `Sound-squatter` and `X-Squatter` showed significantly better performance, with `X-Squatter` achieving perfect coverage and `Sound-squatter` consistently generating nearly all known homophones.

In cross-language homophone coverage, the evaluation focused on the tools' ability to generate homophones for words with identical pronunciations across multiple languages. This scenario included 95 pronunciations with 374 written forms spanning four languages. The findings reveal that `Auto-Squatter` struggled, with a maximum average coverage ratio of only 0.3. Single-language tools like `Sound-skwatter` also performed poorly, as expected, due to their lack of cross-language training. Multi-language models, particularly `X-Squatter`, show the best performance in this task, with `X-Squatter` achieving average coverage of 0.95.

For quasi-homophone generation, the tools' performance was measured using the Weighted Feature Edit Distance, a metric assessing the similarity between pronunci-

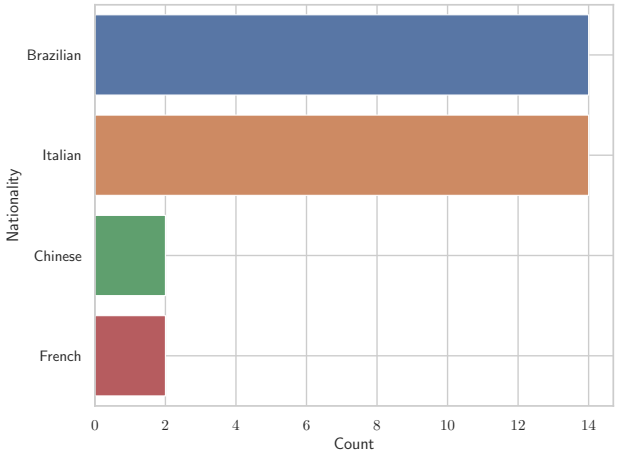
ations. The evaluation involved generating quasi-homophones for 30 English-US words and calculating the distance between target words and generated candidates. The results showed that `Auto-Squatter` had a higher average distance and standard deviation, indicating less accurate generation. Models like `Sound-skwatter` and `Sound-squatter` performed better, but `X-Squatter` demonstrated the most consistent and accurate quasi-homophone generation, with a lower average distance and variability. This model's hybrid approach, incorporating articulatory feature vectors, allows it to produce more accurate quasi-homophones, even in the presence of phonemic representation gaps across languages.

5.2 Can the AI Tools Anticipate People's Mistakes?

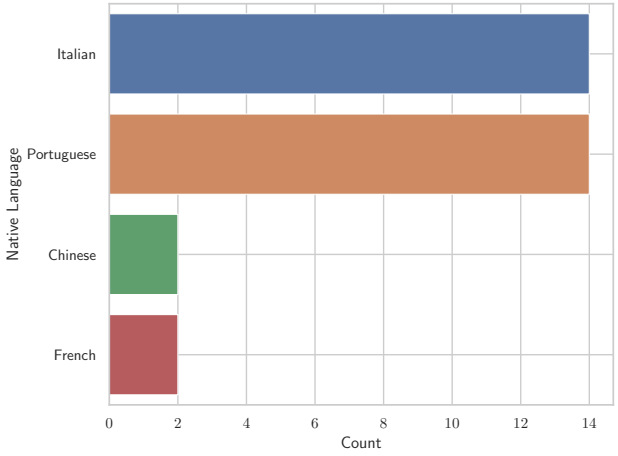
We now present results for the experiment described in Section 3.3. Each participant in the questionnaire is presented with the audio pronunciation of 20 domain names. The participants are then instructed to write the spelling as they understand it. We gathered responses from 32 individuals representing 4 nationalities and speaking 8 languages. The majority of participants is aged between 19 and 40 years and most posses educational qualifications exceeding a Bachelor's degree. Figure 5.5 summarizes the set of users who answered the survey. Most users are native in Portuguese or Italian, with no one native in English. The set of participants has been deliberately selected to focus on young people that are *not* native English speakers, since we are particularly interested in testing cross-language scenarios. Indeed, with this setup we aim to collect samples of mistakes people would face when consuming audio content. Performing a comprehensive study about the frequencies each mistake is observed in practice is left for future work.

No user was able to correctly transcribe all domains, indicating some level of difficulty in the task. Across the 20 domains presented, there were surprisingly 219 different written forms diverging from the actual domain names. The domain with the fewest different transcriptions was `americanexpress.com`, while the one with the most was `centralinvest.ru`. Figure 5.9 illustrates the count of alternative spellings for each domain transcription, highlighting the considerable noise in the task.

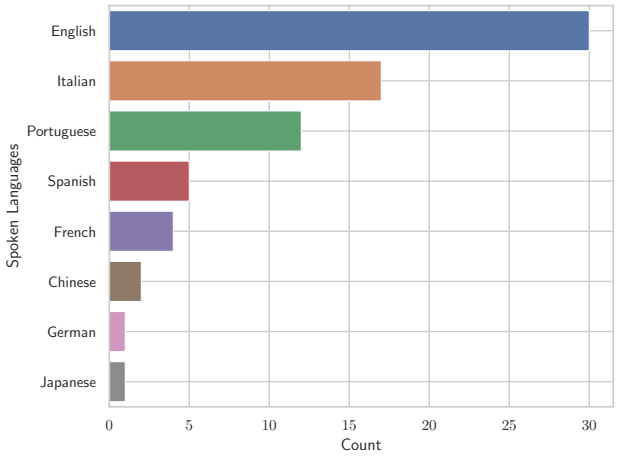
Figure 5.9 also reports valid and invalid domains typed by users. A domain is considered invalid if it does not contain a TLD. The domain `centralinvest.ru` partic-



(a)



(b)



(c)

Fig. 5.5 Users answering the questionnaire reported their (a) nationality, (b) mother-language and (c) other languages they are proficient.

ularly suffered in this aspect, with no user being able to understand the pronunciation of the `.ru` TLD. Some users typed `centrinvesthrough` and `centuryinvestrue`, which approximate the pronunciation but are not valid domains.

Correlating the number of incorrect transcriptions with the popularity of the domain (Rank in the Tranco list) reveals little correlation (0.1558) between the two variables. When considering only valid domains, the correlation is 0.2044, and for only invalid domains, it is -0.1084. Notably, few individuals could correctly spell well-known domains such as **cloudns.com** and **crunchyroll.com**, while most successfully transcribed **cakecentral.com**, which ranks lower in the Tranco ranking.

Table 5.9 This table lists the count of unique domain names written by users for each domain in the questionnaire, compared with the domain's ranking in the Tranco list. The distinction is made between valid and invalid domains, with invalid domains containing either no Top-Level Domain (TLD) or invalid characters.

Tranco Rank	Domain	Invalid	Valid	Total
412	cloudns.net	0	19	19
522	coinmarketcap.com	1	8	9
661	eastday.com	0	8	8
738	crunchyroll.com	0	13	13
741	americanexpress.com	0	2	2
2 732	fsu.edu	1	9	10
5 522	retailmenot.com	0	7	7
5 685	pdst.fm	4	3	7
5 743	awsdns-18.com	4	14	18
7 253	pornbox.com	0	18	18
30 865	nanning.gov.cn	2	23	25
86 722	centrinvest.ru	31	0	31
87 700	cakecentral.com	1	8	9
91 185	playerup.com	0	15	15
103 992	marnet.mk	1	16	17
118 015	sexymasseur.com	1	17	18
474 837	bsta.rs	2	9	11
576 429	uir.ac.id	3	26	29
741 359	kitchenwaresreview.com	2	12	14
964 077	informatica6.com	0	13	13

Although we acknowledge that the investigation needs more individuals and domains for stronger claims, the results suggest that users frequently make spelling mistakes when faced with such tasks. Therefore, caution is needed when referring to locators using voice. It is evident that errors in transcription occur more frequently with domain names that are intrinsically confusing, such as those involving play on words and acronyms. The complete list of responses is provided in Appendix A.

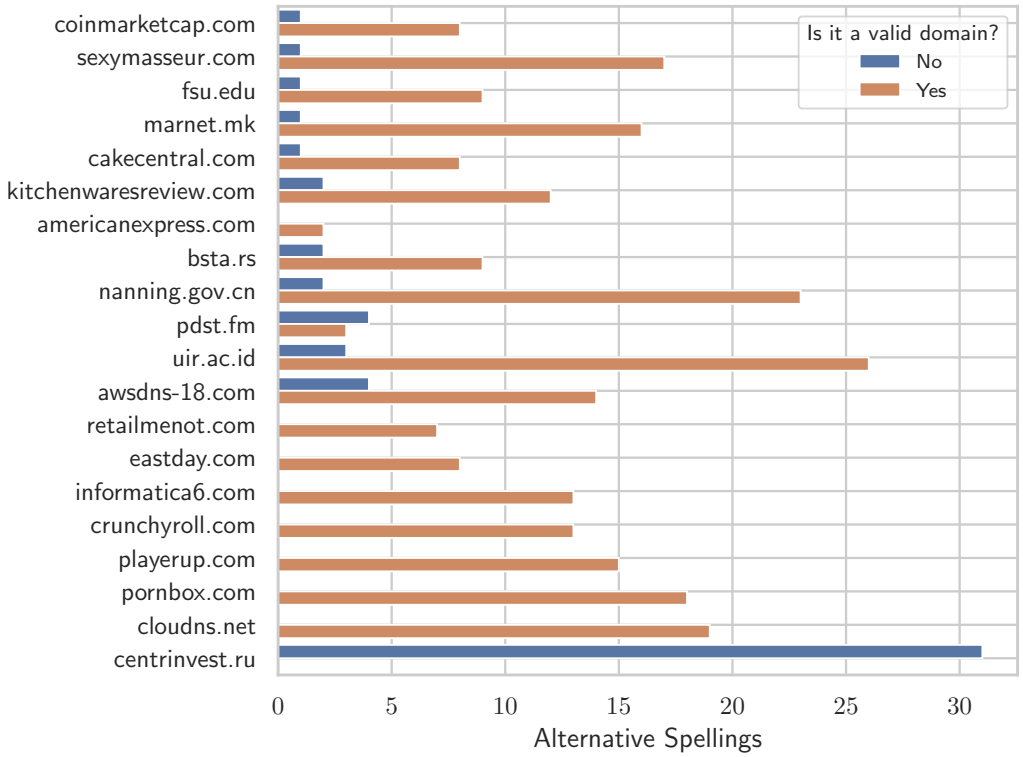


Fig. 5.6 Count of alternative spellings for each domain transcription. The number of alternative spellings suggests that a considerable amount of noise is involved in the task.

5.2.1 Validation via Transcription Errors

Using the obtained list of transcription mistakes, we then check the generated sound-squatting domains. For each of the 20 domains, a list of candidates was generated using all the tools proposed in this thesis and those mentioned in Chapter 2.4, including: `Auto-Squatter`, `Sound-skwatter`, `Sound-squatter`, `X-Squatter`, `DomainFuzz`, `URLCrazy`, `dnstwist`, `URLInsane`, and `AIL`.

Candidates were generated employing different cybersquatting methods: Typo, Homograph, Bit, and Sound-squatting. For sound-squatting, when available, we utilized Read Language and Write Language pairs from every possible combination of English US, English UK, Portuguese Brazilian, and French French.

After generating synthetic candidates, we calculated the intersection between the user's mistakes and the generated candidates. It is important to note that the comparison is done solely on the Second-level domain, as the models proposed in this thesis were not trained to handle full URLs. Therefore, candidates were generated considering only the Second-level domain.

Table 5.10 presents a comparison of the unique candidates produced by various tools for the domains included in the questionnaire. It reveals significant variations in the number of candidates generated by each tool, with data-driven tools such as `Auto-Squatter`, `Sound-skwatter`, `Sound-squatter`, and `X-Squatter` producing relatively fewer candidates compared to third-party tools like `AIL`, `dnstwist`, `DomainFuzz`, `URLCrazy`, and `URLInsane`. While the total number of unique candidates generated by data-driven tools is 8 614, third-party tools collectively produce a substantially larger set of 137 144 unique candidates. However, it's essential to consider the percentage of candidates and mistakes found, where data-driven tools generally exhibit higher percentages compared to third-party tools. This suggests that although data-driven tools generate fewer candidates, they are more effective in identifying potential squatting domains relevant to user transcription mistakes. These results highlight the trade-off between quantity and quality in candidate generation tools.

Additionally, we examined if the tools that generated potential squatting candidates could anticipate the mistakes made by users. Table 5.10 also shows the number of candidates produced by each tool that match user transcription mistakes. The candidates are categorized into "Found" and "Not Found" based on whether

they match user transcription mistakes. Additionally, the total number of candidates ("Total Candidates") and the percentage of candidates ("% Candidates Found") and mistakes found ("% Mistakes Found") are provided.

In total, third-party tools anticipated 53 mistakes, while data-driven tools anticipated 32. These numbers correspond to 24.20% and 14.61% of the wrong transcriptions collected, respectively. The intersection between these two lists is 18. While significant, it is worth noting that third-party tools generate squatting candidates through brute force, whereas data-driven tools found a higher percentage of mistakes with less candidates – that is, the generation has higher quality. This is evident as third-party tools generate at least four times more candidates than our tools. This statement is further supported by the percentage of candidates produced that actually anticipate mistakes: 0.04% for third-party tools and 0.37% for our tools.

Table 5.10 This table displays the counts of unique candidates generated by each tool for the domains included in the questionnaire. The candidates are categorized into "Found" and "Not Found" based on whether they match user transcription mistakes. Additionally, the total number of candidates and the percentage of candidates and mistakes found are provided. The bottom rows represent the total number of unique candidates generated by all data-driven tools and third-party tools, respectively.

Tool	Not Found	Found	Total Candidates	% Candidates Found	% Mistakes Found
`Auto-Squatter`	332	4	336	1.19	1.99
`Sound-skwatter` without Audio	354	4	358	1.12	1.99
`Sound-skwatter`	408	6	414	1.45	2.99
`Sound-squatter`	5 277	19	5 296	0.36	9.45
`X-Squatter`	3 341	21	3 362	0.62	10.45
Total Unique	8 582	32	8 614	0.37	14.61
`AIL`	23 097	45	23 142	0.19	22.39
`dnstwist`	117 812	26	117 838	0.02	12.94
`DomainFuzz`	5 718	29	5 747	0.50	14.43
`URLCrazy`	4 844	30	4 874	0.62	14.93
`URLInsane`	6 315	26	6 341	0.41	12.94
Total Unique	137 091	53	137 144	0.04	24.20

In addition to the raw counts of anticipated mistakes, for the sake of completeness, we calculated precision, recall, and F-Scores for each tool. Precision measures the accuracy of the candidates generated, recall measures the ability to capture all potential mistakes, and the F-Score provides a balance between precision and recall. `X-Squatter` showed a higher recall among our proposals because it effectively generated homograph variants, which were common among user mistakes. On the

other hand, `Auto-Squatter` had a higher precision but lower recall, indicating it generated fewer but more accurate candidates. The complete values are in Table 5.11.

Table 5.11 Performance metrics (Precision, Recall, and F-Score) of various tools in generating squatting candidates. `X-Squatter` demonstrated a higher recall due to its effectiveness in generating homograph variants, which were common among user mistakes. In contrast, `Auto-Squatter` showed higher precision but lower recall, indicating it produced fewer but more accurate candidates.

Tool	Precision	Recall	F-Score
`Auto-Squatter`	5.95e-03	9.13e-03	7.21e-03
`Sound-skwatter` without Audio	5.59e-03	9.13e-03	6.93e-03
`Sound-skwatter`	7.25e-03	1.37e-02	9.48e-03
`Sound-squatter`	2.08e-03	5.02e-02	3.99e-03
`X-Squatter`	3.27e-03	5.02e-02	6.14e-03
`AIL`	9.72e-04	1.03e-01	1.93e-03
`dnstwist`	1.10e-04	5.94e-02	2.20e-04
`DomainFuzz`	2.51e-03	6.62e-02	4.84e-03
`URLCrazy`	3.59e-04	7.08e-02	7.14e-04
`URLInsane`	4.05e-04	5.94e-02	8.04e-04

This study has some limitations. First, the models were trained only on the Second-level domain, which might not fully capture the complexity of URL-based squatting. Second, the tools were evaluated in a controlled environment, which might differ from real-world scenarios. Future work could involve training models to handle full URLs and testing the tools in more diverse and dynamic environments.

5.3 Concluding Remarks on Tool Validation

Each proposed tool builds on the Transformer Networks, with variations in each option to handle different scenarios. Table 5.12 lists the number of trainable parameters for each tool, significantly impacting resource consumption and training time. Since all models utilize the same base architecture (Transformer), the number of parameters primarily influences GPU memory usage. The model sizes are generally similar, except for `Sound-skwatter`, which includes audio signal input. This addition requires extra modules for reconstructing the audio signal via Mel Spectrogram, resulting in a 38 million parameter difference. `Auto-Squatter` is the smallest model, with 7.4 million parameters.

Table 5.12 Model size of the different alternative tools.

Tool	# Trainable Parameters
`Auto-Squatter`	3.7 M + 3.7 M
`Sound-skwatter` without Audio	22.1 M
`Sound-skwatter`	60.2 M
`Sound-squatter`	22.2 M
`X-Squatter`	23.8 M

Since each proposed tool evolves based on feedback and limitations of the previous one, Table 5.13 provides a comprehensive summary of the data-driven models in terms of their capabilities. The evaluated aspects include Quality Control, Data-driven Homophone Generation, Multiple Homophone Generation, Multi-language Support, and Cross-language Support, all essential considering the research questions we stated.

Table 5.13 Summary of proposed data-driven tools and their capabilities.

Tool	Quality Control	Data-driven homophone generation	Multiple homophone generation	Multi-language Support	Cross-language Support
`Auto-Squatter`	No	Yes	Yes	No	No
`Sound-skwatter`	Yes	Yes	Yes	No	No
`Sound-squatter`	Yes	Yes	Yes	Yes	Partial
`X-Squatter`	Yes	Yes	Yes	Yes	Yes

To analyze the redundancy and coverage of squatting techniques among the tools listed in Table 5.14, we consider the same 20 domains evaluated previously. Subsequently, we generated candidate domains using each alternative and compared the sets of unique names produced by each tool. Table 5.14 compares the amount of unique candidates generate by each tool for the 20 domains. While the quality of the data-driven generation is discussed in the previous section, here, it focuses solely on the set of sound-squatting candidates without quality concerns. There is considerable variation in the number of candidates generated by each alternative. For instance, `dnstwist` stands out by producing over 117 836 names, most of which are unique, with approximately 81.05% being exclusive to the `dnstwist` set.

Particularly interesting to cite is the significant redundancy observed among candidates generated by the other alternatives, as depicted in Table 5.15. It's important to note that third-party tools share a significant portion of the candidates among each other, while the ratio is considerably smaller when compared to our tools.

Table 5.14 Tools for squatting candidate generation and evaluation.

Tool	**Techniques**	**Unique**
`Auto-Squatter`	Sound	336
`Sound-skwatter` without Audio	Sound	358
`Sound-skwatter`	Sound	414
`Sound-squatter`	Sound	5 289
`X-Squatter`	Sound	3 357
`AIL`	Typo, Homograph, Bit,	23 141
`dnstwist`	Typo, Homograph, Bit	117 836
`DomainFuzz`	Typo	5 738
`URLCrazy`	Typo, Homograph, Bit, Sound	2 788
`URLInsane`	Typo, Homograph, Bit, Sound	4 523

Table 5.15 Intersection in the generation of third-party tools for squatting candidate generation and our proposal. The intersection represents the proportion of shared candidates between the set of candidates in the rows and the set of candidates in the columns, normalized by the size of the set in the row.

	`Auto-Squatter`	`Sound-skwatter` without Audio	`Sound-skwatter`	`Sound-squatter`	`X-Squatter`	`AIL`	`dnstwist`	`DomainFuzz`	`URLCrazy`	`URLInsane`
`Auto-Squatter`	1.000	0.048	0.051	0.054	0.071	0.062	0.068	0.065	0.060	0.065
`Sound-skwatter` without Audio	0.045	1.000	0.288	0.285	0.257	0.148	0.145	0.145	0.131	0.131
`Sound-skwatter`	0.041	0.249	1.000	0.297	0.263	0.133	0.133	0.133	0.126	0.130
`Sound-squatter`	0.003	0.019	0.023	1.000	0.116	0.044	0.039	0.041	0.023	0.025
`X-Squatter`	0.007	0.027	0.032	0.182	1.000	0.102	0.070	0.072	0.041	0.052
`AIL`	0.001	0.002	0.002	0.010	0.015	1.000	0.221	0.219	0.097	0.168
`dnstwist`	0.000	0.000	0.000	0.002	0.002	0.043	1.000	0.040	0.019	0.026
`DomainFuzz`	0.004	0.009	0.010	0.038	0.042	0.882	0.828	1.000	0.388	0.632
`URLCrazy`	0.007	0.017	0.019	0.044	0.049	0.807	0.804	0.800	1.000	0.755
`URLInsane`	0.005	0.010	0.012	0.029	0.038	0.858	0.675	0.802	0.465	1.000

At this point, it is possible to speculate that sound-squatting is a feasible technique for cybersquatting, as users often make mistakes while transcribing locators that can be anticipated by third-party tools. However, the extensive coverage of third-party tools is partly due to the large number of candidates they generate. The data-driven tools we propose are less prolific but more specialized, capable of anticipating mistakes at a similar level as these third-party tools with a lower percentage of useless candidates.

Chapter 6

Potential Applications

To explore the potential application of a generative model for sound-squatting proactive search, this chapter delves into two specific scenarios: domain squatting checking within TLS certificates and PyPI packages.

Firstly, we compile a comprehensive list of popular locators and generate their homophones and quasi-homophones across various languages. Subsequently, we search for these identified candidates within the aforementioned contexts. We focus on two primary resources: PyPI packages (in Section 6.1) and domain names present in registered TLS certificates (in Section 6.2). In both cases, besides generating sound-squatting candidates, we employ alternative squatting tools to generate additional candidates, enabling a comparative analysis between sound-squatting and other squatting methodologies.

As a baseline, we utilize the AIL tool, which employs a suite of 21 algorithms including typo-squatting, homograph-squatting, and list-based sound-squatting. For simplicity, candidates generated by AIL are referred to as "other-squatting".

6.1 Squatting on PyPI Repository

The investigation starts by analyzing squatting opportunities on the Python Package Index (PyPI). This involves identifying packages on the PyPI platform that could potentially be used for squatting popular packages. Characteristics and statistics of these packages are examined, comparing sound-squatting against the baseline. It is

important to note that verifying the actual maliciousness of all these candidates is beyond the scope of this study. Instead, the focus is on checking attack opportunities and highlighting differences between sound-squatting candidates and other squatting methods.

The investigation begins by compiling a list of the 5,000 most downloaded packages in the PyPI repository during the 30 days preceding June 20, 2023. The ranking data is gathered from the `PyPI Stats API` [68], providing package-related statistics up to the last 120 days. Following this, candidate package names are generated using both sound-squatting and other-squatting techniques. The repository index is then queried to identify the occurrence of these candidate packages. The analysis spans a duration of 967 days, from November 20, 2020, to July 14, 2023. For this specific use case, `X-Squatter` parameters are configured to limit the number of candidates to $k = 35$ per domain, with a threshold of $p = 0.9999$ and a temperature of $t = 1.0$.

In total, a substantial number of candidate package names are generated. Specifically, 522 042 sound-squatting candidates and 7 675 548 other-squatting candidates are created for 5 000 packages. This results in approximately 15 times more other-squatting candidates per project, which is expected given the diverse range of algorithms implemented for other-squatting generation.

Table 6.1 provides an overview of the number of candidates generated and the total number of candidates that exist on the platform at some point within the 967-day span. Additionally, the intersection between the techniques shows that sound-squatting is not fully covered by the other algorithms. The groups are shown in Figure 6.1.

Table 6.1 Pairs of candidates and projects found online per technique, along with the intersection comprising candidates matching both sound-squatting and other-squatting techniques.

Technique	Not Found	Found	Total
Other-squatting	(1) 7 632 674	(4) 21 691	7654365
Sound-squatting	(2) 499 838	(5) 1 021	500859
Sound-squatting and Other-squatting	(3) 20 390	(6) 793	21183
Total	8152902	23505	8176407

The amount of other-squatting candidates is ≈ 14.70 times the amount of sound-squatting candidates. Yet, the percent of candidates found to exist on the platform

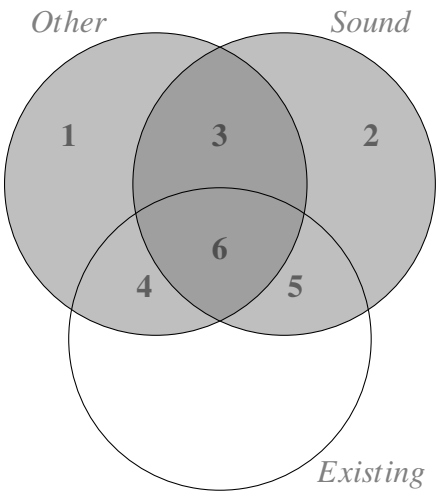


Fig. 6.1 Venn Diagram representing the data. Gray areas represents the data reported at Table 6.1. Counts are not shown to improve readability.

over the total of generated for other-squatting is $\approx 0.29\%$ while the percentage increases to $\approx 0.35\%$ for sound-squatting. It is also interesting to check the intersection of the pairs of candidates (target, candidate) that are produced by both techniques. Candidates found in both lists have a much higher chance of being found on the platform: approximately 3.89% of candidates are indeed found to exist at some point during the period.

Figure 6.2 shows the cumulative count of found candidates per day, divided by the total number of generated candidates, this time the pairs target package and candidate is not considered to avoid counting multiple times the same candidate that is valid for multiple packages, in consequence, only the set of candidates is considered. Separated lines show sound-squatting candidates (yellow) and other-squatting (blue).

In the temporal analysis, we note a particular behavior starting on Feb 2021. To improve its analysis, we show the ratio of packages found online per day by the number of targets (5 000) in Figure 6.3. Naturally, other-squatting is above sound-squatting in this figure as we have more other-squatting then sound-squatting candidates.

On Feb 12, 2021, a total of 573 packages that are other-squatting generated candidates were created in the platform. We see in Figure 6.3 a major increase in the ratio of other-squatting candidates found online, followed by a sharp drop. Between Feb 25, 2021, and Feb 26, 2021, approximately 73 000 packages were suspended in

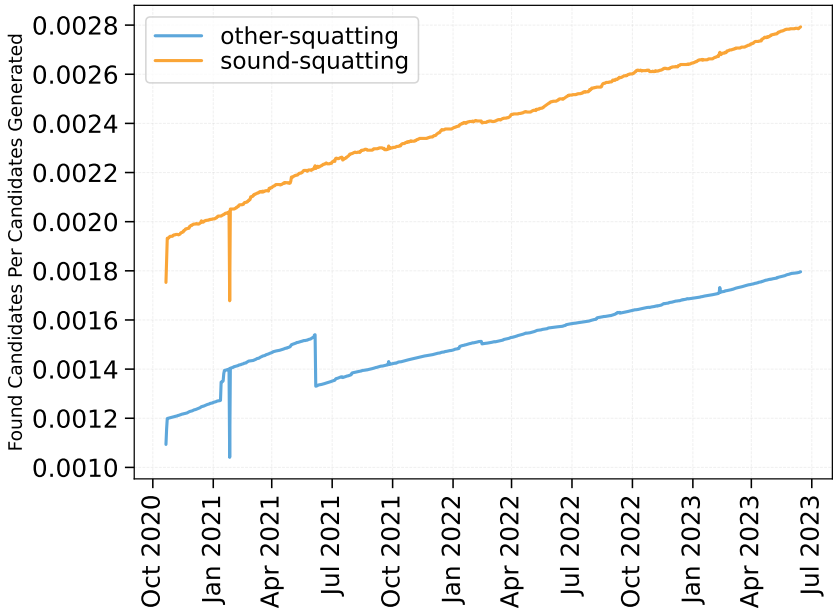


Fig. 6.2 Found candidates over total generated candidates.

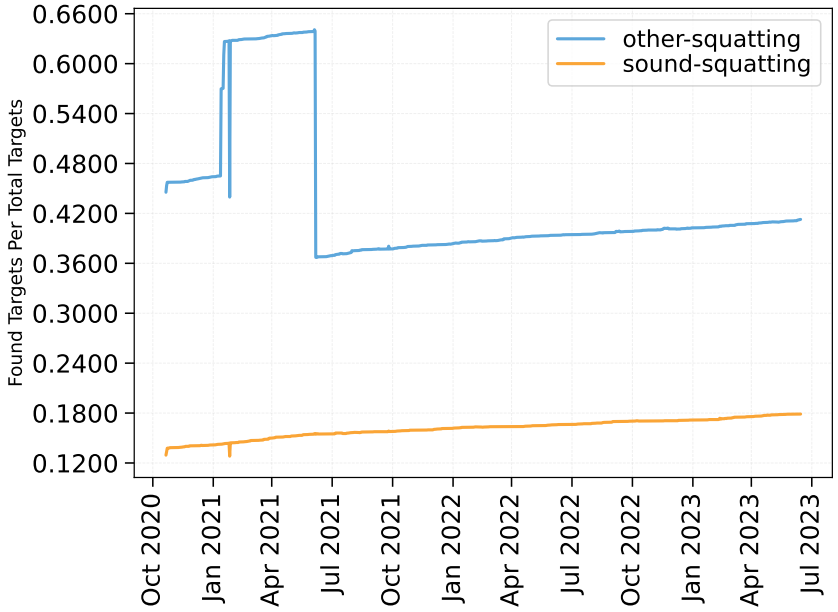


Fig. 6.3 Found candidates over the number of evaluated targets.

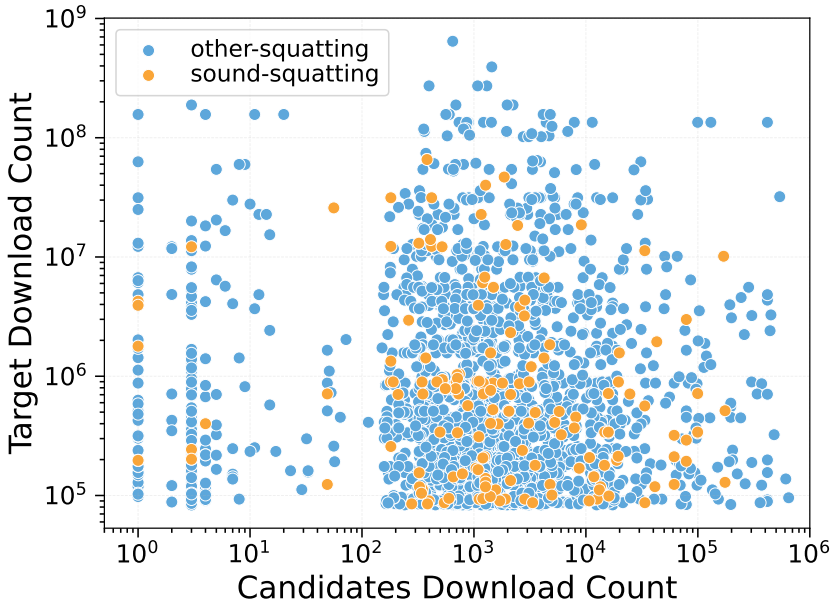


Fig. 6.4 Downloads of candidates vs. downloads of target packages.

the PyPI platform due to violations of the platform rules. In the aftermath, on Feb 27, a large portion of the suspended packages were reinstated (see the plateau after that day in Figure 6.3), as not all removed packages could be linked to malicious activity.

Finally, on July 05, 2021, 1 610 packages were permanently deleted from PyPI, with a substantial number also present in list of other-squatting candidates, as shown by the sharp drop for other-squatting in Figure 6.3. This event has been documented in reputable sources.[1]

Figure 6.4 displays a scatter plot of the number of downloads of the package targets (y-axis) versus the number of downloads of the candidates found online (x-axis). Note the different scales. Each point marks a candidate and colors represent sound-squatting (yellow) and other-squatting (blue). Sound-squatting candidates are found to have download statistics comparable to other-squatting types. Note, in particular, that many sound-squatting candidates present large download counts.

We, finally, perform a manual qualitative analysis of some sound-squatting candidates generated by `X-Squatter`, presenting in Table 6.2 salient examples. Some of these packages are periodically deleted by the platform and republished

[1] See: https://www.theregister.com/2021/03/02/python_pypi_purges/ and https://github.com/pypi/support/issues/935

right after the removal. Not all of them are malicious. In particular, `requestes` is a package that warns users against installing packages without checking them first. Most packages have a low reputation with almost no documentation, which indicates a parking or squatting attempt. One striking example is `pirec`, which is very similar in pronunciation to the legitimate package `pyrect`. However, `pirec` is a software that can execute shell scripts, giving attackers a high degree of freedom once victims install the package.

Table 6.2 Examples of candidates generated by `X-Squatter` and found online in the PyPI repository.

Squatting	Legitimate Package	Context
scrapi	Scrapy	Empty repository
sfinx	sphinx	Low reputation project without documentation.
requestes	requests	Warning page
regex	rejex	Affiliated project (legitimate).
skema	schema	Low reputation project without documentation.
flasque	flask	Low reputation project without documentation.
noompy	numpy	Project uses similarity as a form of homage.
pidantic	pydantic	Low reputation project without documentation.
pirec	pyrect	Unrelated. Squatting project, allow shell command execution.

Our investigation in the previous chapter reveals that users commonly make mistakes during the transcription of domain names, some of which can be anticipated using homophone generation methodology. Taken together, these results suggest that sound-squatting represents a feasible technique for malicious actors seeking to propagate malware via PyPI. We believe that `X-Squatter` can aid PyPI's moderators in implementing measures to effectively mitigate the potential risks associated with the use of sound-squatting techniques.

6.2 Domain Impersonation

This section moves into domain squatting, particularly within the context of domains utilizing HTTPS connections. Squatting on domains that use HTTPS connections poses a significant threat, as valid certificates can deceive users into perceiving the squatted domain as legitimate [10]. Similar to the previous use case, our focus here is on comparing sound-squatting candidates with other types of squatting candidates. While we provide some examples of malicious sound-squatted domains, conducting

a comprehensive verification of maliciousness falls outside the scope of our work. Our emphasis in this verification lies on the second-level domain, which narrows down the search scope. Examining the second-level domain allows us to focus on entities actively seeking to acquire misleading domains, while subdomains present a more expansive and potentially unlimited landscape.

Attackers are known to register TLS certificates for squatted domains to increase the success rate of phishing campaigns [10]. We search for squatting candidates using registered domains found in certificates that we collected via the `CertStream` (Certificate Transparency Logs) [12]. We select the top 10 000 most accessed domains from the Tranco Ranking List [46] (accessed on July 10, 2023). Since the success of the phishing website depends on the existence of a valid TLS certificate, the issuance of the certificate happens very early in the life-cycle of the phishing campaign.

Using `X-Squatter`, we generate a total of 552 143 sound-squatting candidates for 10 000 second-level domains, employing several cross-language configurations using all trained languages (see Table 4.5). In this use case, we set `X-Squatter` parameters to limit the number of candidates to 35 per domain, and use as a threshold $p = 0.9999$ and a temperature of $t = 1.0$, which were the values that rendered move coverage in the results from Section 5.1. Again, we generate candidates for other-squatting candidates with AIL. We obtain 8 177 376 other-squatting candidates for the same 10 000 second-level domains, and, among those, 34 268 candidates generated for both techniques given the same domain target.

We collected certificates spanning 124 days, from February 11, 2023, to June 14, 2023. In total, we gathered 866 125 758 certificates representing 127 775 910 domains across 5 155 Top-Level Domains (TLDs). These certificates originated from a total of 263 certificate issuers located in 50 different countries.

Using the `CertStream` data, we extract all server names to build a comprehensive list of registered second-level domains. We match the generated candidates with the second-level domains extracted from `CertStream` data.

Table 6.3 details the total number of candidates and those we found in `CertStream`. We notice a trend similar to the PyPI use case, however with much higher percentages. While $\approx 10,4\%$ of other-squatting candidates have a registered TLS certificates, this percentage increases to $\approx 13,1\%$ for sound-squatting. Names found in both lists

Table 6.3 Pairs of Target Domain and Candidates generated and registered domains per technique, along with the intersection of pairs matching both sound-squatting and other-squatting techniques.

Technique	Not Registered	Registered	Total
Other-squatting	7 293 770	849 338	8 143 108
Sound-squatting	445 382	72 493	517 875
Sound-squatting and Other-squatting	17 946	16 322	34 268
Total	7 757 098	938 153	8 695 251

again have a much higher chance of being found on `CertStream`: $\approx 47,6\%$ of generated candidates are present in at least one TLS certificate.

To observe how the phenomenon evolves over time, Figure 6.5 presents the proportion of registered candidates we identified existing compared to the total number of domains found in all certificates issued on each respective day. Note how the ratio is constant over time, with some minor spikes in particular days. That is, the registration of potentially abused domains is very high and continuous. Naturally, other-squatting is above sound-squatting as we have much more other-squatting candidates.

Figure 6.6 provides insight into the relationship between TLDs and sound-squatting candidates. Note, the TLD `.io` stands out with an average of 467.46 (standard deviation 146.94) unique registered domains that match our candidates per day, a share of 3.98%. To complement the picture, in Figure 6.7, we represent the position of the most popular TLDs according to the number of registered domains and their ranking position based on the number of squatting candidates per TLD. Ideally, we would expect a linear relationship around the red line, indicating that the number of squatting candidates is proportional to the number of domains managed by each TLD. This is indeed the general trend. However, we also observe some outliers in the figure.

Among the top 100 most used TLDs, which sum to almost 95% of all domain names in the period collected and also reduce the noise in the analysis, we find that TLDs such as `.io`, `.app`, and `.dev` deviate significantly from the diagonal line. Even more concerning is the outlier TLD `.us`. These findings align with recent reports of phishing campaigns hosted in the `.us` TLD [2].

[2]https://it.slashdot.org/story/23/09/02/1415234/why-are-godaddys-us-domains-being-used-for-so-much-phishing

In Figure 6.7, there are also interesting examples such as `.tk` and `.ml`, which exhibit a smaller ratio of squatting to the total number of registered domains. This variation among TLDs in their susceptibility to squatting activities highlights the need for a close look into each TLD's security and potential misuse.

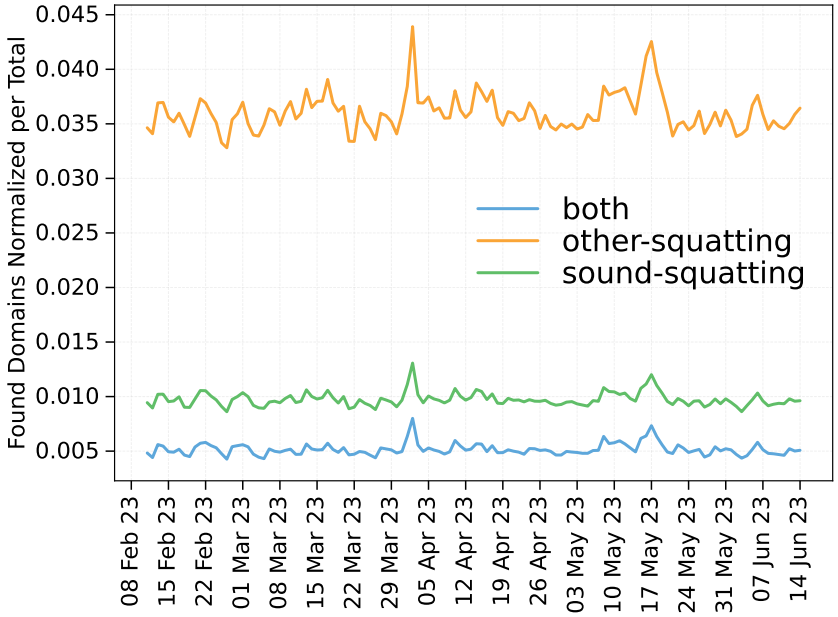


Fig. 6.5 Registered candidates over total registered domains on issued certificates per day.

6.2.1 Manual Qualitative Analysis

To assess the effectiveness of our generation, we chose 12 second level domains extracted from domains with known histories of phishing attacks. Specifically, we choose: amazon, bankofamerica, dropbox, facebook, icloud, instagram, linkedin, microsoft, netflix, paypal, steamcommunity, and tripadvisor for our analysis.

Subsequently, we obtain the candidates using `X-Squatter` and conduct a manual verification process to assess whether these domains exhibited phishing characteristics. The manual verification covers 1 038 domains. We use the tool Puppeteer to capture screenshots, setting a timeout of two seconds.

We organize the candidates into nine distinct classes that effectively capture legitimate/abuse characteristics. This classification scheme aids in gaining insights into the nature and potential purposes behind these domains:

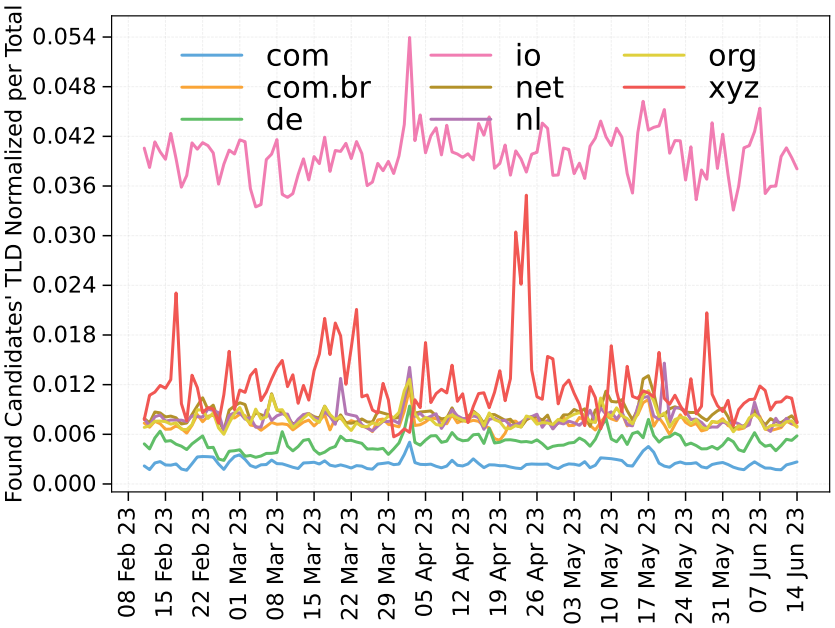


Fig. 6.6 Ratio of registered sound-squatting candidates per TLD.

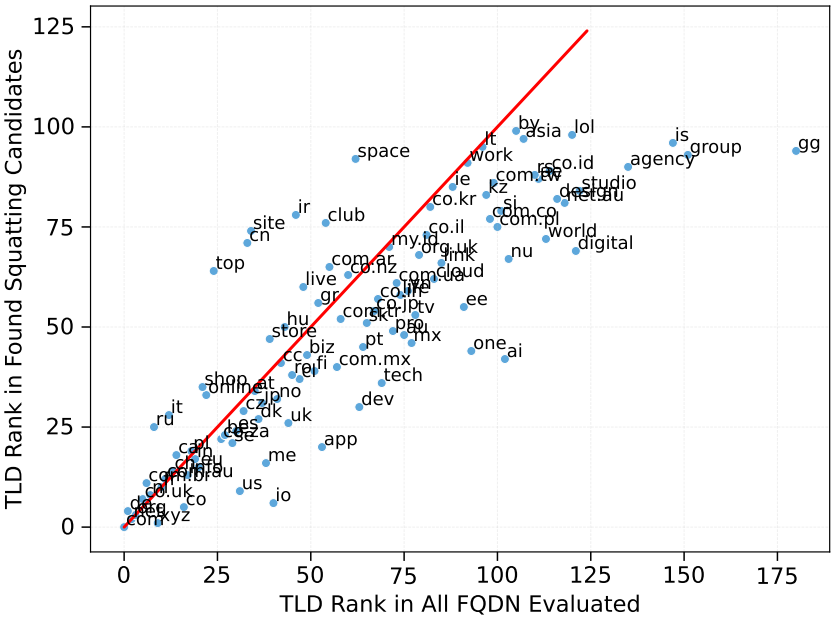


Fig. 6.7 Relationship of TLD and squatting candidates.

- **Error Domains (463 domains):** The "Error Domains" category consists of domains that do not present any content and are shown as errors in the browser. These domains likely represent typographical errors, misconfigurations, or inactive websites. While they may not be malicious, they still have relevance because phishing domains present periods of inactivity during they life-cycle [45], therefore a registered domains with an error page might indicate one of this stages.

- **Legitimate Domains (279):** These are genuine, non-squatting domains that are not involved in any malicious or deceptive activities of notice. They serve their intended purpose without any evidence of fraud.

- **Domains for Sale (117):** Domains in this category are put up for sale. While not necessarily malicious, they might be squatting on potentially valuable domain names with the intention of selling them at a higher price.

- **Parked Domains (90):** Parked domains are placeholders typically used by domain owners or registrars. They often display advertisements or a generic landing page, and their primary purpose is to generate revenue through ad clicks.

- **Redirector Domains (22):** Redirector domains are used to redirect web traffic from one domain to another. They can be legitimate, but they may also be used in various online scams and malicious campaigns.

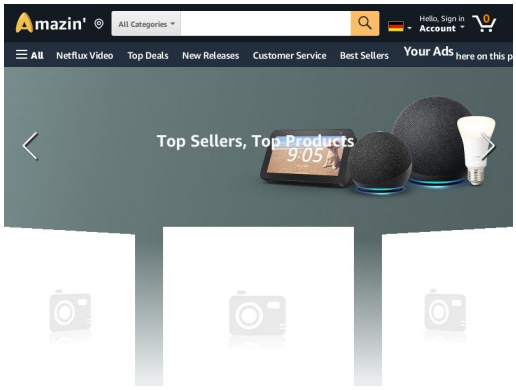
- **Authoritative Owned Domains (18)** Some brands chose to buy problematic domains and redirect traffic to their services. These are called Authoritative Owned domains. We discovered these domains by checking DNS data obtained from Whois with the information from the target domain which the candidates was generated.

- **Hit-Stealing Domains (16):** Hit-stealing domains might be involved in schemes where they try to intercept or steal web traffic intended for other legitimate domains, potentially for fraudulent purposes. The domains names are usually not similar at all with the website name or brand associated.

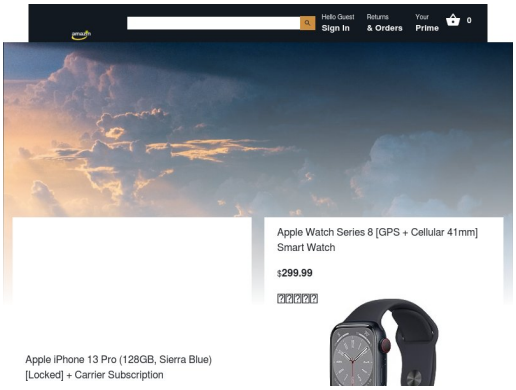
- **Look-Alike Domains (10):** Look-alike domains often resemble well-known, legitimate pages, but with slight variations that might go unnoticed. They are typically used in phishing or deception attempts.

- **Phishing Campaign (7):** Domains categorized under this label are "Look-Alike Domains" likely part of phishing campaigns, where they are used to trick users into revealing sensitive information or credentials.
- **Other (16):** This category includes domains that do not fit neatly into the previous defined categories or require further investigation to determine their purpose and intent.

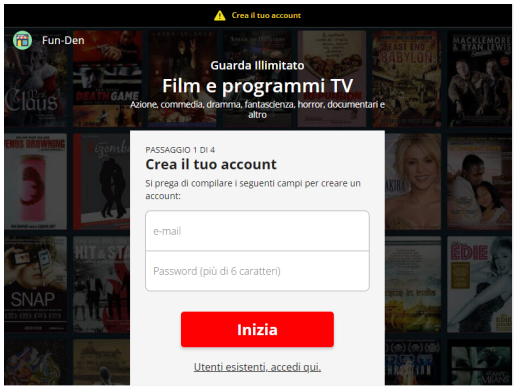
We show in Figure 6.8 some examples of phishing domains found during the classification.



(a) Phishing Amazon



(b) Phishing Amazon



(c) Phishing Netflix

Fig. 6.8 Phishing examples for Amazon.com and Netflix.com found in selected candidate domains.

Interestingly, some brands chose to buy problematic domains and redirect traffic to their services. We call these Authoritative Owned domains. Focusing on the Redirector Domains, we discovered by checking DNS data obtained from Whois that 18 quasi-homophone domains for Amazon and Apple are Authoritative Owned.

Four domains are owned by an organization responsible for brand protection. Eight domains belong to third-party organizations and for 10 domains we could not find any information regarding ownership.

Chapter 7

Final Considerations and Conclusions

This chapter compares the models, summarizes their strengths and weaknesses, describes the implications of the thesis, and outlines its limitations.

7.1 Linguistic Coverage

Languages within the Indo-European family, specifically English (United States and Great Britain), French (France), Portuguese (Brazil), Italian (Italy), and Spanish (Spain), were mainly used in the training and in the use cases. These languages collectively represent a substantial portion of online users and resources [47]. While this choice helped to set an extensive evaluation of phonetic aspects due to the availability of tools and resources, it resulted in the exclusion of important languages from other language families such as Russian, Chinese, and Arabic.

The proposed methodology for multi-language, as presented in `Sound-squatter` and `X-Squatter`, can be extended to other languages. The International Phonetic Alphabet (IPA) offers a universal framework, enabling the generation of Feature Vectors for other languages. This flexibility was demonstrated by reporting results for quasi-homophone generation. Adapting *Grapheme to Phoneme* to the input language and adjusting a specific *Grapheme Decoder* module to the written forms of the output language via fine-tuning or retraining are necessary steps for applying these models to other languages.

Regarding adaptability, a potential challenge arises if the rules of *Grapheme to Phoneme* are no longer applicable, which would occur if new words are added to a language, e.g., due to the natural evolution of languages. However, languages often adapt spellings and pronunciations slowly, making this scenario rare. Moreover, words borrowed from one language to another typically maintain a similar pronunciation but change in written form.

A limitation in the methodology is its lack of handling of heteronyms, i.e., words with the same written form but different pronunciations, common in non-phonetic languages like English and French. This is primarily a problem to be addressed in the *Grapheme to Phoneme* module. The eSpeak NG is designed to produce a single pronunciation alternative and does not cover heteronyms. Properly handling them requires using a *Grapheme to Phoneme* that generates all alternative pronunciations for the same word. Measuring the practical impact of the lack of this feature is challenging, as their classification depends on context.

7.2 Scalability

The models can be trained in a single commodity server within a few hours. Compared to list-based alternatives, more computational resources are required during training. However, this results in improved quality and coverage of the generated candidates.

Different Transformer architectures were compared during the design of the tools, but more recent alternatives, particularly those based on generative AI, were not considered. These Large Language Models (LLMs) are characterized by their large number of parameters [13]: for example, Bert [17] has 100 million parameters, GPT-2 has 1.5 billion parameters [48], and LLama2 has 65 billion parameters [62]. Typically, specializing these models to specific problems requires fine-tuning, thus significant effort in terms of data collection and computational resources. Instead, a decision was made to construct a specialized model to address a domain-specific problem, avoiding the fine-tuning of large or huge models. This approach enables a reduction in model size while enhancing performance in the specific task. `X-Squatter`, for instance, has $22M$ parameters and leverages architectural choices finely crafted for the homophone generation problem.

7.3 Conclusions

In this thesis, we addressed the problem of generating sound-squatting candidates by proposing a comprehensive methodology that can handle variations across single-language and cross-language scenarios. We adopted a data-driven approach to bypass the need for fixed rules allowing our models to generalize effectively across diverse linguistic contexts.

Our investigation centered on three research questions:

1. How does data-driven machine learning facilitate generalization across diverse sound-squatting scenarios and linguistic contexts?

We developed multiple sound-squatting generation tools, starting with a naive model and progressing to a sophisticated architecture based on the Transformer Neural Network. Through ablation studies, we identified essential components for effective sound-squatting generation. Our approach considered various data modalities and encoding strategies, ensuring robust performance across different scenarios. Quantitative and qualitative evaluations confirmed our models' ability to generate homophones and quasi-homophones with high coverage and quality.

2. Can AI-based sound-squatting generation replicate or anticipate mistakes users make during transcription?

To verify if our models can anticipate user transcription mistakes, we designed a questionnaire to gather data on common errors made during domain name transcription. By comparing these errors with the outputs of our models, we assessed their ability to predict user mistakes. Our findings indicated that the models could indeed replicate a portion of these mistakes, quantifying their practical utility.

3. Are sound-squatting candidates produced by our methodology found in practical use cases?

We investigated the presence of sound-squatting candidates in real-world scenarios by cross-referencing our generated candidates with existing domain names and Python software packages. This analysis revealed the practical applicability of our methodology and its potential to identify instances of misuse or suspicious cases.

Our work establishes a foundation for future research on sound-squatting and its implications across various linguistic contexts and application domains. Future

work will include the deployment of the techniques to monitor possible abuse of sound-squatting in other scenarios.

References

[1] (n.d.). Amazon mechanical turk. Accessed on June 7, 2024.

[2] (n.d.). Clickworker. Accessed on June 7, 2024.

[3] (n.d.). Google Text-to-Speech. Accessed on: 2024-05-05.

[4] Agten, P., Joosen, W., Piessens, F., and Nikiforakis, N. (2015). Seven months' worth of mistakes: A longitudinal study of typosquatting abuse. In *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS 2015)*. Internet Society.

[5] AIL-PyPI Squatting (2023). Ail- pypi squatting. https://github.com/typosquatter/pypi-squatting. Accessed on: 09/08/2023.

[6] Association, T. I. P. (2022). The international phonetic association homepage. https://www.internationalphoneticassociation.org/. Accessed on 2022-10-05.

[7] Atkinson, K. (2006). Gnu aspell 0.60. 4.

[8] ATT&CK, M. (2022a). Capec-616: Establish rogue location. https://capec.mitre.org/data/definitions/616.html.

[9] ATT&CK, M. (2022b). Capec-631: Soundsquatting. https://capec.mitre.org/data/definitions/631.html.

[10] Awad, M., Allam, A. E., Salameh, K., and Mazrouei, R. A. (2022). Phishing for legitimacy: The use of ssl certificates to ensnare internet users. In *2022 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, pages 313–317.

[11] Bureau., B. B. (Accessed: 2024). Bbb scam alert: Using voice search? use caution when asking for auto dial from a smart device. https://www.bbb.org/article/news-releases/20523-scam-alert-using-voice-search-use-caution-when-asking-for-auto-dial-from-your-smart-device

[12] (CaliDog), S. Certstream. https://certstream.calidog.io/. [Accessed 08-05-2024].

[13] Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P. S., Yang, Q., and Xie, X. (2024). A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.* Just Accepted.

[14] Chollet, F. et al. (2015). Keras. https://github.com/fchollet/keras.

[15] Crystal, D. (2008). *Dictionary of linguistics and phonetics*. John Wiley & Sons.

[16] (DARPA), D. A. R. P. A. Arpabet. A dialect of the International Phonetic Alphabet (IPA) designed for American English speech synthesis. Referenced from https://huggingface.co/spaces/Harveenchadha/oiTrans/resolve/main/indic_nlp_resources/script/arpabet.pdf. Accessed: May 28, 2024.

[17] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

[18] Dinaburg, A. (2011). Bitsquatting: Dns hijacking without exploitation. *Proceedings of BlackHat Security*.

[19] dnscan (2023). dnstwister: Dns twist web service. https://dnstwister.report/. Accessed on: 09/08/2023.

[20] Doherty, L. (2022). ipa-dict - monolingual wordlists with pronunciation information in IPA. https://github.com/open-dict-data/ipa-dict.

[21] DomainFuzz (2023). DomainFuzz: A domain name permutation tool. https://github.com/monkeym4ster/DomainFuzz. Accessed on: 09/08/2023.

[22] Dryer, M. S. and Haspelmath, M., editors (2013). *WALS Online (v2020.3)*. Zenodo.

[23] Duddington, J., Avison, M., Dunn, R., and Vitolins, V. (Accessed: 2023). espeak ng text-to-speech. https://github.com/espeak-ng/espeak-ng.

[24] Durumeric, Z., Kasten, J., Bailey, M., and Halderman, J. A. (2013). Analysis of the HTTPS certificate ecosystem. In *Proceedings of the 2013 conference on Internet measurement conference*. ACM.

[25] Foundation, P. S. (2018). Invalid projects.

[26] Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 369–376, New York, NY, USA. Association for Computing Machinery.

[27] Gu, Y., Ying, L., Pu, Y., Hu, X., Chai, H., Wang, R., Gao, X., and Duan, H. (2023). Investigating package related security threats in software registries. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 1578–1595.

[28] Holgers, T., Watson, D. E., and Gribble, S. D. (2006). Cutting through the confusion: A measurement study of homograph attacks. In *USENIX Annual Technical Conference, General Track*, pages 261–266.

[29] Khan, M. T., Huo, X., Li, Z., and Kanich, C. (2015). Every second counts: Quantifying the negative externalities of cybercrime via typosquatting. In *2015 IEEE Symposium on Security and Privacy*, pages 135–150. IEEE.

[30] Kim, D., Cho, H., Kwon, Y., Doupé, A., Son, S., Ahn, G.-J., and Dumitras, T. (2021). Security analysis on practices of certificate authorities in the HTTPS phishing ecosystem. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. ACM.

[31] Kintis, P., Miramirkhani, N., Lever, C., Chen, Y., Romero-Gómez, R., Pitropakis, N., Nikiforakis, N., and Antonakakis, M. (2017). Hiding in plain sight: A longitudinal study of combosquatting abuse. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 569–586.

[32] Kumar, D., Paccagnella, R., Murley, P., Hennenfent, E., Mason, J., Bates, A., and Bailey, M. (2018). Skill squatting attacks on amazon alexa. In *27th USENIX security symposium (USENIX Security 18)*, pages 33–47.

[33] Littell, P., Mortensen, D. R., Lin, K., Kairis, K., Turner, C., and Levin, L. (2017). URIEL and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 8–14, Valencia, Spain. Association for Computational Linguistics.

[34] Liu, X., Duh, K., Liu, L., and Gao, J. (2020). Very deep transformers for neural machine translation. *arXiv preprint arXiv:2008.07772*.

[35] LLC, G. (2024). Google topics api. https://developers.google.com/search/docs/topics/. Accessed: May 29, 2024.

[36] Lodge, D. (2023). URLCrazy: Domain name permutation and availability checker. http://www.morningstarsecurity.com/research/urlcrazy. Accessed on 09/08/2023.

[37] Loyola, P., Gajananan, K., Kitahara, H., Watanabe, Y., and Satoh, F. (2020). Automating domain squatting detection using representation learning. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1021–1030.

[38] Marshall, C. C., Goguladinne, P. S., Maheshwari, M., Sathe, A., and Shipman, F. M. (2023). Who broke amazon mechanical turk? an analysis of crowdsourcing data quality over time. In *Proceedings of the 15th ACM Web Science Conference 2023*, WebSci '23, page 335–345, New York, NY, USA. Association for Computing Machinery.

[39] Michaelis, S. M., Maurer, P., Haspelmath, M., and Huber, M., editors (2013). *APiCS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

[40] Moran, S. and McCloy, D., editors (2019). *PHOIBLE 2.0*. Max Planck Institute for the Science of Human History, Jena.

[41] Mortensen, D. R., Dalmia, S., and Littell, P. (2018). Epitran: Precision g2p for many languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

[42] Mortensen, D. R., Littell, P., Bharadwaj, A., Goyal, K., Dyer, C., and Levin, L. (2016). PanPhon: A resource for mapping IPA segments to articulatory feature vectors. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3475–3484, Osaka, Japan. The COLING 2016 Organizing Committee.

[43] Nikiforakis, N., Balduzzi, M., Desmet, L., Piessens, F., and Joosen, W. (2014). Soundsquatting: Uncovering the use of homophones in domain squatting. In *International Conference on Information Security*, pages 291–308. Springer.

[44] Nikiforakis, N., Van Acker, S., Meert, W., Desmet, L., Piessens, F., and Joosen, W. (2013). Bitsquatting: Exploiting bit-flips for fun, or profit? In *Proceedings of the 22nd international conference on World Wide Web*, pages 989–998.

[45] Oest, A., Zhang, P., Wardman, B., Nunes, E., Burgis, J., Zand, A., Thomas, K., Doupé, A., and Ahn, G.-J. (2020). Sunrise to sunset: Analyzing the end-to-end life cycle and effectiveness of phishing attacks at scale. In *29th {USENIX$}$ Security Symposium (${$USENIX$}$ Security 20)*.

[46] Pochat, V. L., Van Goethem, T., Tajalizadehkhoob, S., Korczyński, M., and Joosen, W. (2018). Tranco: A research-oriented top sites ranking hardened against manipulation. *arXiv preprint arXiv:1806.01156*.

[47] Q-Success Web-based Services (2024). Usage statistics of content languages for websites.

[48] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

[49] Ren, Y., Ruan, Y., Tan, X., Qin, T., Zhao, S., Zhao, Z., and Liu, T.-Y. (2019). Fastspeech: Fast, robust and controllable text to speech.

[50] Roberts, R., Goldschlag, Y., Walter, R., Chung, T., Mislove, A., and Levin, D. (2019). You are who you appear to be. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. ACM.

[51] Semmlow, J. (2018). Chapter 6 - linear systems in the frequency domain: The transfer function. In Semmlow, J., editor, *Circuits, Signals and Systems for Bioengineers (Third Edition)*, Biomedical Engineering, pages 245–294. Academic Press, third edition edition.

[52] Sonowal, G. (2020). A model for detecting sounds-alike phishing email contents for persons with visual impairments. In *2020 Sixth International Conference on e-Learning (econf)*, pages 17–21. IEEE.

[53] Sonowal, G. and Kuppusamy, K. (2019). Mmsphid: A phoneme based phishing verification model for persons with visual impairments. information and computer security journal.

[54] Spaulding, J., Nyang, D., and Mohaisen, A. (2017). Understanding the effectiveness of typosquatting techniques. In *Proceedings of the Fifth ACM/IEEE Workshop on Hot Topics in Web Systems and Technologies*, HotWeb '17, New York, NY, USA. Association for Computing Machinery.

[55] Staib, M., Teh, T. H., Torresquintero, A., Mohan, D. S. R., Foglianti, L., Lenain, R., and Gao, J. (2020). Phonological features for 0-shot multilingual speech synthesis. *arXiv preprint arXiv:2008.04107*.

[56] Stevens, K. N. (1998). *Acoustic phonetics*. MIT press.

[57] Szurdi, J., Kocso, B., Cseh, G., Spring, J., Felegyhazi, M., and Kanich, C. (2014a). The long {"Taile"$}$ of typosquatting domain names. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 191–206.

[58] Szurdi, J., Kocso, B., Cseh, G., Spring, J., Felegyhazi, M., and Kanich, C. (2014b). The long "Taile" of typosquatting domain names. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 191–206, San Diego, CA. USENIX Association.

[59] Tenney, I., Das, D., and Pavlick, E. (2019). BERT rediscovers the classical NLP pipeline. *arXiv preprint arXiv:1905.05950*.

[60] Tian, K., Jan, S. T. K., Hu, H., Yao, D., and Wang, G. (2018). Needle in a haystack: Tracking down elite phishing domains in the wild. In *Proceedings of the Internet Measurement Conference 2018*, IMC '18, page 429–442, New York, NY, USA. Association for Computing Machinery.

[61] Torroledo, I., Camacho, L. D., and Bahnsen, A. C. (2018). Hunting malicious TLS certificates with deep neural networks. In *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*. ACM.

[62] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

[63] (UCL), U. C. L. Sampa (speech assessment methods phonetic alphabet). https://www.phon.ucl.ac.uk/home/sampa/. Accessed: May 28, 2024.

[64] Valentim, R., Drago, I., Cerutti, F., and Mellia, M. (2022). Ai-based sound-squatting attack made possible. In *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 448–453. IEEE.

[65] Valentim, R., Drago, I., Cerutti, F., and Mellia, M. (2023a). Sound-skwatter (did you mean: Sound-squatter?) ai-powered generator for phishing prevention. *arXiv preprint*.

[66] Valentim, R., Drago, I., Cerutti, F., and Mellia, M. (2024). X-squatter: Ai multilingual generation of cross-language sound-squatting. In *ACM Transactions on Privacy and Systems (TOPS)*.

[67] Valentim, R., Drago, I., Mellia, M., and Cerutti, F. (2023b). Lost in translation: Ai-based generator of cross-language sound-squatting. In *2023 IEEE European Symposium on Security and Privacy Workshops (EuroSPW)*, pages 513–520.

[68] van Kemenade, H. and Erdin, E. (2019). hugovk/pypistats 0.4.0.

[69] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[70] Veyseh, A. P. B., Lai, V., Dernoncourt, F., and Nguyen, T. H. (2021). Unleash GPT-2 power for event detection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 6271–6282.

[71] Wang, Y.-M., Beck, D., Wang, J., Verbowski, C., and Daniels, B. (2006). Strider typo-patrol: Discovery and analysis of systematic typo-squatting. *SRUTI*, 6(31-36):2–2.

[72] Yang, Y.-Y., Hira, M., Ni, Z., Chourdia, A., Astafurov, A., Chen, C., Yeh, C.-F., Puhrsch, C., Pollack, D., Genzel, D., Greenberg, D., Yang, E. Z., Lian, J., Mahadeokar, J., Hwang, J., Chen, J., Goldsborough, P., Roy, P., Narenthiran, S., Watanabe, S., Chintala, S., Quenneville-Bélair, V., and Shi, Y. (2021). Torchaudio: Building blocks for audio and speech processing. *arXiv preprint arXiv:2110.15018*.

[73] Zeng, Y., Zang, T., Zhang, Y., Chen, X., and Wang, Y. (2019). A comprehensive measurement study of domain-squatting abuse. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pages 1–6.

[74] Zhang, N., Mi, X., Feng, X., Wang, X., Tian, Y., and Qian, F. (2019). Dangerous skills: Understanding and mitigating security risks of voice-controlled third-party functions on virtual personal assistant systems. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1381–1396. IEEE.

Appendix A

Mistakes in Transcriptions Collected from Questionnaire Responses

Table A.1 List of domains and their transcriptions - Part 1 of 2

Domain	Set of User Transcriptions
americanexpress.com	americainexpress.com, americanexpress.com
awsdns-18.com	ewsdnf-eighteen.com, iwsdns-18.com, awsdns.18.com, awlsdns-18.com, ewsdns-18.com, lwsdns-18.com, awusdns-18.com, awsdns_eighteen.com, awsdns-eighteen.com, awstns-eighteen.com, awsdns18.com, awstcnn.com, awsdns-18.com, ewsdnf-18.com, aws-sdns-18.com, awstdns-18.com, awsdns_18.com, euusdns 18.com
bsta.rs	bsta.ers, psta.rs, bsta.os, bst8doctors, psta.ors, vsta.rs, vsta.ors, bsta.us, bstars, bsta.rs, bsti.rs
cakecentral.com	8central.com, katecentral.com, keycentral.com, cakecentral.com, cakescentral.com, kcentury.com, capecentral.com, catecentral.com, cake central.com
centrinvest.ru	central invest tree, centreinvesttrought, centreinvestrue, centralinvest3, centralinvestry, centre invest tree, centroinvestro, centralvestral, centruminvestry, centuryinvestrue, centerinvestdream, santruminvestrough, centerinvesture, santronvastree, centrancewestrue, center investory, centerinvestyou, centreinvestry, centerinvestrew, central investee, centreinvestru, centralinvestrue, saintinvestry, centryvestree, centerinvesthrough, centerinvesttrough, centrinvestrue, century inventory, cent invest you, centrinvesthrough, sentryfasttrue
cloudns.net	clounds.net, claudents.net, cloudense.net, clouden.net, cloudandsto.net, clouddense.net, cloudins.net, cloudings.net, cloudns.net, clouds.net, cloud.net, cloudends.net, cloud-en.net, claudance.net, cloudens.net, cloudin.net, cloudance.net, cloudans.net, couldin.net

Continue on next page.

Table A.2 List of domains and their transcriptions - Part 2 of 2

Domain	User Transcription
coinmarketcap.com	coinmarket.com, coinmarketcat.com, coinmarketcup.com, cointmarketcap.com, coinmarketcapital com, coinmarketcamp.com, coinmarketcap.com, cleanmarketcap.com, toymartketcat.com
crunchyroll.com	crounchyroll.com, crumchirow.com, crunchiro.com, crunchyword.com, crunchyroll.com, crouchyroad.com, crunchero.com, chrunchyrole.com, crunchroll.com, crunchyrow.com, crunchyroad.com, creamshire.com, crancyroad.com
eastday.com	eastaid.com, yesterday.com, esatedate.com, eastdate.com, eastday.com, isday.com, istday.com, eaststate.com
fsu.edu	fsu.ddu, fseu.edu, afcdutu, fsu.edu, fsu.tdu, fscu.edu, fse.edu, fsu.idu, fsu.idy, fsu.idyou
informatica6.com	informaticalsix.com, infomatica6.com, informaticalsixt.com, informaticassist.com, informaticcursics.com, informatica6.com, informaticsix.com, informaticus6.com, informatiquesix.com, informatic6.com, informatics6.com, informaticasix.com, informatical6.com
kitchenwaresreview.com	kitchenwearsreviews.com, kitchenwheresreview.com, com, kitchenwestreview.com, kitchenwaresreview, kitchenswearreview.com, kitchenwhereasreviews.com, kitchenwaresreview.com, kitchenwearsreview.com, kitchenresreview.com, kitchenwearsreveal.com, kitchenwarereviews.com, kitchenwesreview.com, kitchenusreview.com
marnet.mk	manet.mk, monets.mk, management.k, mynet.mk, manette.key, munad.mk, monet.mk, marnets.mk, manart.mk, monet.uk, manot.mk, mermaid.mk, monet.nk, manner.mk, mannet.mk, manned.mk, marnet.emkey
nanning.gov.cn	naming.cn, 9dogcnn, namen.gov.cn, nanen.gov.cn, noning.gov.cn, nunnin.gov.cn, naming.gap.cn, namen.got.cm, naning.god.cn, noning.gap.cn, nanning.gov.cn, nanin.gov.cn, notname.gov.cn, naming.gov.cn, nanning.catdog.cn, lamen.gov.cn, nanning.com.cn, mining.gov.cn, nan.gov.cn, naning.gov.cn, nanning.biaf.cn, nine.gov.cn, nannin.gov.cn, nanning.cn
pdst.fm	pdctfm, pdstfm, ptsd.fm, pdst.fm, pdstfn, pdftfm, pds.fm
playerup.com	player.com, plerap.com, flareup.com, playrop.com, play.com, pleura.com, playrot.com, playeurope.com, playwrap.com, playarp.com, playrup.com, playerup.com, playrap.com, playup.com, playart.com
pornbox.com	puwnvox.com, pullanbox.com, pumbox.com, pointbox.com, pawnvocks.com, poombox.com, homebox.com, honbox.com, cornebox.com, vulnbox.com, toonbox.com, punbox.com, coonbox.com, pumdocs.com, pornbox.com, pwnbox.com, pawnbox.com, poonbox.com
retailmenot.com	retailmeno.com, retailme.com, retailmino.com, retellmeno.com, ratailmenot.com, retailmenot.com, retellmenot.com
sexymasseur.com	sexymasseur.com, sexymassed.com, sexseemassert.com, sexymesser.com, sexymessy.com, sexymasee.com, sexymaster.com, sexy.com, sexymassert.com, sexymeser.com, sexymicer.com, sexymassage.com, sexmace.com, sexymassa.com, sexymasset.com, sexmassage.com, sexymasser.com
uir.ac.id	uir.hak.id, ura.ack.id, uyr.tac.id, ura.act.id, uar.tak.id, ula.tag.id, uar.act.id, uael.ak.id, uir.act.id, url.ak.id, uri.hack.id, uar.hack.id, uea.tak.id, url.act.id, uri.hack.it, uio.tac.id, uir.ac.id, ui.tak.id, uilo.hack.id, ual.act.id, uaeat.id, uardoctadoctad, uilr.hack.id, uir.x.id, uir.ack.id, uar.a.at, youre.id, uil.tac.id