INVESTIGATING PROPERTIES OF PHONOTACTIC KNOWLEDGE THROUGH WEB-BASED EXPERIMENTATION

A Dissertation Presented

by

PRESLEY PIZZO

INVESTIGATING PROPERTIES OF PHONOTACTIC KNOWLEDGE THROUGH WEB-BASED EXPERIMENTATION

A Dissertation Presented

by

PRESLEY PIZZO

Approved as to style and content by:

_____
Joe Pater, Chair

_____
Kristine Yu, Member

_____
Brian Dillon, Member

_____
Emery Berger, Member

_____
John Kingston, Department Chair
Linguistics

DEDICATION

For the impostors.

ACKNOWLEDGMENTS

I am grateful to many people for their help and support while I was writing this dissertation.

As I started to write about Joe Pater, I wondered how many hours he has spent discussing phonology with me. I settled on "too many to count." Joe's generosity with his time — no matter what country he happens to inhabit — has provided a continuous thread throughout my graduate career, despite the many changes in course I took. His flexibility and breadth allowed me to explore and settle on a path that makes sense for me, while still maintaining a focus on phonology. Joe struck an impressive balance as an advisor, giving me the space to make my own decisions while working hard to carve out opportunities for me. He has supported me regardless of my career choices, and that made the difference in my ability to write this dissertation.

Kristine Yu also dedicated an immense amount of time to helping me think through ideas and experimental designs. Her broad knowledge base and incisive thinking were extremely helpful in both brainstorming and refining the experimental chapters of this dissertation. What I will remember more, though, is how her genuineness and interest in others made her so approachable, giving me the confidence to ask her a wider variety of questions and opening up opportunities to learn practical things about how to organize your research. Her quirky sense of humor made the long road more enjoyable.

Brian Dillon was an indispensable source of advice. Meetings with Brian are like no other meetings I had in grad school; they feel like untangling a knot. Perhaps this is because he brings a novel perspective as someone outside of my subfield; I prefer the hypothesis that he is extraordinary.

Emery Berger has an inspirational vision for programs that can solve real problems. He and Emma Tosch have done incredible work on SurveyMan, the precursor to the program that I present in this dissertation. Although I had toyed with the idea of writing experiment-running software, I likely never would have done so without their work paving the way. I am indebted to them for their ideas and the time they spent with me as we discussed goals for experimental software. I am also grateful to Emery for taking the time to coach me during my job search. As a linguist going into a career in software, I had little formal guidance, and his advice helped me understand the task I was undertaking and succeed in it.

John Vilk also played a role in making Speriment possible, by recommending TypeScript to Emma and me as an alternative to JavaScript. If this sounds like a minor contribution, I leave it as an exercise to the reader to make sense of the JavaScript type system.

I'm grateful to Will Dabney for grabbing my computer one day in a cafe and showing me how to write object oriented code. Having someone I could ask dumb questions made it possible for me to make the jump from being able to get a computer to do what I want it to do (somehow, eventually), to being able to design software.

Wendell Kimper was the first person I met at UMass, and he quickly became a friend and mentor. Wendell taught me how to navigate grad school; Brian Smith gave me new ways of looking at theories; Claire Moore-Cantwell was endlessly patient with both my linguistic musings and my personal conundrums; Claire, Robert Staubs, and Minta Elsman — the phonologists in my cohort — were extremely generous with their ideas when I shared my work with them. The rest of my cohort, Elizabeth Bogal-Allbritten, Seda Kan, Jason Overfelt, and Andrew Weir, helped me through first-year syntax, and as a whole my class was a welcoming, supportive group that I'm grateful to be a part of. I'm also proud of our trivia team, Totally Jake, and its multiple wins against the likes of The Cunning Linguists.

I was lucky that the phonology department at UMass has had a strong tradition of informal meetings throughout my time here, whether in the form of grant group meetings, reading groups, or seminars. I thank Lena Fainleib, Clint Hartzell, Ivy Hauser, Coral Hughto, Karen Jesney, John Kingston, John McCarthy, Kevin Mullin, Aleksei Nazarov, Amanda Rysling, and all the visitors and occasional drop-ins we've had over the years for their insightful presentations and helpful comments.

I'd also like to thank all of the friends who created a support network for me throughout the academic and personal challenges of grad school. Northampton was a change for me but it turned out to be a wonderful one. On that note, I'd also like to thank all of the cafes in Northampton for, well, keeping me going.

Mike MacHenry not only encouraged me to become a better software developer and helped me do so, but also talked me through every interview and every p-value greater than 0.05. Everyone should have someone in their corner who only says "I told you so" after a victory. He was also a great sounding board as I designed Speriment, both because of his software development expertise and his background in designing a survey language.

My family has always been in full support of me, no matter what I choose to do. I feel so lucky to have always been confident that I could make the career choices I wanted without fear of disappointing my family, or of annoying them by answering "Still writing my dissertation" every time they ask me what I'm up to. Thank you.

ABSTRACT

INVESTIGATING PROPERTIES OF PHONOTACTIC KNOWLEDGE THROUGH WEB-BASED EXPERIMENTATION

SEPTEMBER 2015

PRESLEY PIZZO

B.A., EMORY UNIVERSITY

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Joe Pater

The goal of this dissertation is to advance the state of the art of research in constraint-based phonotactics. It takes a two-pronged approach: a technological contribution intended to facilitate future research, and experiments which seek to shed light on high-level questions about the properties of phonotactic models that can guide the development of theoretical work.

The technological contribution is a software package called Speriment which allows experimenters to create and run experiments over the internet without advanced programming techniques. This software is particularly well suited to the kinds of experiments often run in phonotactic research, but can also be used for experiments in other domains of linguistics and the social sciences. It is hoped that this software will make it faster and easier to conduct phonotactic and other experiments as well as encourage experimenters to increase the reproducibility and transparency of their research.

The experiments presented here address questions that assume constraint-based phonotactic frameworks, but that do not rely on particular theories of the content of the constraint set. That is, they apply to constraint-based frameworks for theories of phonotactics, with the first study seeking to distinguish between two such frameworks, a linear version of Harmonic Grammar and Maximum Entropy, while the second investigates whether phonotactic knowledge is independent of knowledge of phonological alternations. These coarse-grained questions about phonotactic knowledge on how pieces of phonotactic knowledge interact with each other and with another part of the grammar are intended to add to the groundwork on which phonotactic models and models of all phonological knowledge are built. Their findings have implications for which constraint-based frameworks should be used for future theories and how these theories can be reliably tested.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

The goal of this dissertation is to advance the state of the art of research in constraint-based phonotactics. It takes a two-pronged approach: a technological contribution intended to facilitate future research, and experiments which seek to shed light on high-level questions about the properties of phonotactic models that can guide the development of theoretical work.

The technological contribution is a software package called Speriment which allows experimenters to create and run experiments over the internet. This software is particularly well suited to the kinds of experiments often run in phonotactic research, but can also be used for experiments in other domains of linguistics and the social sciences. It is hoped that this software will make it faster and easier to conduct phonotactic and other experiments as well as encourage experimenters to increase the reproducibility and transparency of their research.

The experiments presented here address questions that assume constraint-based phonotactic frameworks, but that do not rely on particular theories of the content of the constraint set. The first study seeks to distinguish between two such frameworks, a linear version of Harmonic Grammar and Maximum Entropy Grammar, while the second investigates whether phonotactic knowledge is independent of knowledge of phonological alternations. These coarse-grained questions about phonotactic knowledge — how pieces of phonotactic knowledge interact with each other and with another part of the grammar — are intended to add to the groundwork on which phonotactic models and models of all phonological knowledge are built. Their findings have implications for which constraint-based frameworks should be used for future theories and how these theories can be reliably tested.

The usefulness of experimentation in improving our understanding of phonotactic knowledge is central to this dissertation. Although the value of experimentation is largely recognized within the field of phonology, it is informative to look at the ways in which it advances our understanding of the questions addressed here. Even when calling into question the need for syntacticians to conduct experiments rather than informally collecting intuitions, Phillips (2009) points out that experiments are useful in cases where questions are too subtle to be satisfactorily answered by introspection. Experiments 1 and 2 in Chapter 2 are useful in demonstrating exactly how such cases can arise.

An intuition in phonology often takes the form of a 'yes' or 'no' answer to whether a word is acceptable in a given language. I will call these *first-degree intuitions*. Graded first-degree intuitions are also possible, describing the degree to which a word is acceptable. Given two graded first-degree intuitions, we can ask which word is the more acceptable of the two: a *second-degree intuition*. In order to address the question in Chapter 2 of how constraint violations combine, we need *third-degree intuitions*: a comparison between second-degree intuitions which are themselves a comparison between first-degree intuitions. Linguists disagree about the consistency of first-degree intuitions in syntax (Gibson and Fedorenko, 2013). Second- and third-degree intuitions are even more difficult to access. So in addition to the other virtues of experiments described by Gibson and Fedorenko, we should recognize that experimental designs and quantitative methods allow us to use first-degree intuitions to build answers to second- or third-degree questions. Instead of eliciting third-degree intuitions, we can elicit all of the requisite first-degree intuitions and compare them using quantitative methods. These methods allow us to do the comparison externally, across participants and items instead of within one person's mind.

Chapter 3 requires experimentation for a different reason: it applies different treatments to different groups in order to look for a causal relationship between treatment

and subsequent behavior. While again eliciting only first-degree intuitions, those intuitions stand to be affected by the training period. The only way to draw conclusions about the effect of training is to employ controls, by using two treatments and looking for a difference between their outcomes. This a common use for experiments in all sciences, and in phonological research is particularly crucial to artificial language learning experiments.

For these reasons, experiments are well-suited to the questions addressed here as well as many others in phonological theory, and tools to facilitate their use are likely to remain in high demand. This dissertation explores ways to more efficiently create and share experiments and their results, as well as ways of using experiments to probe the structure of phonotactic knowledge.

Chapter 1 discusses the benefits of web-based experimentation specifically, how Speriment compares to other experiment creation software packages, and how Speriment is used. Chapter 2 gives a review of literature that suggests that the effects of multiple phonotactic constraint violations accumulate in a word, so that the grammaticality of a word depends on all of the violations it contains. I present two experiments with the ability to refine this statement by investigating how the accumulation is calculated, weighing on the question of whether linear Harmonic Grammar or Maximum Entropy Grammar is a better model for phonotactics. In Chapter 3, I consider whether learning a phonological alternation increases the weight of the constraint motivating that alternation in the phonotactic grammar. Theories differ in whether they couple alternation-based and phonotactic knowledge, with conflicting evidence from typology. I present two experiments that look for an effect of alternations on phonotactics using an artificial language learning paradigm.

CHAPTER 1

SPERIMENT

1.1 Introduction

Speriment is software that facilitates the design and running of experiments over the internet. It's built to work with psiTurk (McDonnell et al., 2012), a program that runs arbitrary JavaScript experiments on Mechanical Turk and other platforms. psiTurk takes much of the work out of interfacing with Mechanical Turk and managing participants, and Speriment further reduces the workload of the experimenter. Instead of writing a dynamic website in JavaScript, Speriment users can write a description of the structure of their experiment in Python, and the JavaScript will be generated automatically.

1.2 Benefits of Online Experiments

Web-based experiments are not applicable to all experimental designs and participant pools, but for those experiments that can be run online, evidence is building that they are a useful and valid tool. Keller et al. (2009a), Sprouse (2011), and Crump et al. (2013) tested results of experiments run over the internet against known results from lab studies, and found that for many tasks, even those dependent on measuring reaction times, online studies replicate known effects. Many of the reasons to consider running experiments online are well-known. The makers of WebExp (Keller et al., 2009b), for instance, note that the internet is a large source of potential participants, and that experiments conducted online avoid scheduling issues and other sources of overhead. There is a further benefit of running experiments online that follows from

the way it makes experiments so quick and relatively inexpensive to run, which is that it supports iterative development, in which experiments are tested multiple times and edited along the way. This is helpful for both practical and scientific reasons.

For practical purposes, online experiments can be treated as user-facing software and can follow parts of the workflow that commercial software products follow. As the software industry is large and competitive, much thought has been put into maximizing productivity in both writing software and responding to customer behavior. Some of the resulting ideas can be applied to the experimental design and deployment process.

One lesson that has been learned in the software industry has been to shift from the "waterfall" model (Royce, 1970) of development to an iterative model (see for instance Agile Alliance (2001)). The waterfall model is one in which each stage of development is thoroughly completed before the next stage is started. It is advantageous when the final stage is extremely costly; better to spend extra time ensuring previous stages are perfect than to risk a mistake in the expensive last stage. It applies well to the manufacture of hardware, and to experiments run in the lab, where procuring participants is time-consuming and costly, and the number of available participants is limited.

In contrast, iterative development is a model in which there are many cycles of development and testing, minimizing the importance of foresight by replacing it with hindsight. This model is effective when testing the product is not very expensive and editing prior work is less time-consuming than starting from scratch, so that the ability to benefit from new information outweighs the cost of revisiting stages of work. It is difficult to predict the perfect design for a product or experiment, and difficult to detect the kinds of problems that a varied user base will have with either a commercial software product or an academic experiment, so responsiveness to feedback can be more useful than extensive prior planning.

Iterative development has proven applicable to experimentation in this work; for both of the studies presented here, I ran several small pilots in quick succession before settling on a design. Each successive pilot was planned and implemented in response to results and problems from the last, and often multiple pilots were possible in one day. Upon reflecting on the results from Experiments 1 and 3, weaknesses in their designs became apparent, and I was able to adjust and rerun them to produce Experiments 2 and 4 in a short period of time.

The scientific contribution of iteratively developed experiments is the ability to increasingly use pilots as a tool for avoiding wasted resources when experiments don't go as planned. The social sciences have struggled with several challenges to experimental validity which are difficult to overcome as long as they are encouraged by the structure of our research.

One of these problems is data-peeking, in which experimenters check interim results for statistical significance and decide whether to run more participants based on the outcome. This practice increases the probability of Type I error; an experimenter who finds a significant effect at the $\alpha = 0.05$ level after engaging in data-peeking has a greater than 5% chance that the effect is due to random chance (Armitage et al., 1969; McCarroll et al., 1992; Strube, 2006). This practice is well-known in medical studies, where it is of ethical importance not to continue ineffective or harmful treatments, but John et al. (2012) shows that it also takes place in psychology. There are ways to correct for this increase in Type I error, rendering the practice a valid statistical technique (Pocock, 1977; Todd et al., 2001; Sagarin et al., 2014). Although linguists would do well to adopt this correction when applicable, it is not currently in widespread use in our field. The ability to easily rerun an experiment in a matter of hours for a relatively low price may be, at least in the short term, more effective at changing habits than the availability of new statistical methods.

Another problem is that of running multiple experiments in search of the same effect and only publishing the one that produced a significant result, referred to as publication bias. Again, this increases the chance of Type I error (Sterling, 1959). The true solution to this problem is the publication of null results, and the journal *Cancer Epidemiology Biomarkers & Prevention* has started a section for null results for exactly this reason (Shields, 2000). However, this kind of change requires cooperation from both journals and the researchers submitting to them, so once again, any change that is more attainable in the meantime would be beneficial.

The ability to write online experiments as open-source computer programs, the text of which is accessible to the researcher, opens up the possibility of making the entire experimental process transparent with the use of version control on a public site such as GitHub. Version control is a process that is easily applicable to any plain text, including the experimental design scripts used to run online experiments. It keeps track of the revisions to a program or other document over time, so that previous versions can be restored if a problem is found with the current one. This is convenient for one's personal design process, but it also allows the concept of reproducibility of research to be applied not just to the final version of an experiment, but to all versions. Researchers may not want to share their results with the world before they have finalized a journal submission, but it is possible to build up version control information in private and publish it to a public repository when the time is right. If this practice were adopted, it would be easier to reason about the true probability that a result was achieved by chance in light of the number of similar attempts that were made. In addition, replication of not only the final experiment but also intermediate versions would be possible. Speriment and the program it works with, psiTurk (McDonnell et al., 2012), make it simple to replicate the analyses as well, as they keep all user data in a database. This database makes it easy to prevent the same participant from taking an experiment more than once, or from taking more

than one version of the same experiment. As long as the database table assigned to the experiment is the same, participants who are already represented in the database will not be allowed to take the experiment. Thus, the most convenient route — saving data from all versions of an experiment in one place — is also the one that maximizes transparency and replicability.

A further long-term goal of Speriment is to encourage experimenters to commit to the criteria by which they will exclude participants before running an experiment, and then doing this exclusion automatically. It is difficult to predict all potential relevant criteria, and so once again the ability to be responsive to information gleaned from pilots will be crucial to the adoption of this feature. If adopted, it would provide additional reassurance that the experimenter had not tried every way of analyzing the data before settling on a way that gave desirable results.

1.3   Comparison with Other Frameworks

There are many other frameworks for creating online experiments, all with different features, strengths, and weaknesses. Many of their differences come down to how they cope with the trade-off between ease of use and expressive power. At one end of the spectrum is a completely expressive system: a Turing-complete programming language. An experimenter who prioritizes flexibility above all can simply write an entire dynamic website in JavaScript and a server in the language of their choice. This requires extensive knowledge of programming and a considerable time investment. Even for experienced programmers, writing each experiment from scratch would waste time that could be saved by reusing encapsulated code that performs certain common features. This motivates the use of frameworks for experiment writing, but still leaves open the question of how many features should be encapsulated.

On the less expressive extreme of the spectrum would be a survey creation website intended for casual users. It would make a few tasks very easy to express, and the

rest impossible to express. Experimenters frequently find that the features of very simple tools are insufficient to test their hypotheses properly.

Experiments will always need to serve a website, randomize item order, display text, and log data, but beyond that, the lists of commonly used and rarely used features vary from field to field and even from experimenter to experimenter. Similarly, different experiment designers vary in the amount of time they wish to spend programming an experiment and the variety of experimental designs they use. As a result, a wide variety of frameworks, different in the number and type of features they make simple to use, have been found useful. After introducing Speriment in terms of its placement on this spectrum and key features, I will compare it to some of the frameworks used in linguistics and related fields.

1.3.1 Properties of Speriment

The guiding philosophy behind Speriment is that experiments common in phonology should be simple to express, and that an experimenter need not know how more advanced designs are expressed in order to describe a more basic one. This philosophy requires the encapsulation of quite complex features, such as different distributions of items and blocks across participants and per-participant page and option construction. In order to make it easy to express these complex designs, and even easier to write experiments that don't need them, Speriment provides several high-level features that can be used by supplying the name of the feature and, when necessary, a data structure of related information. Features that aren't needed simply aren't named, and the defaults are set up to make it safe for experimenters to ignore the documentation on features they don't use.

This prioritization means that some kinds of experiments are difficult or impossible to express. Some of these kinds of experiments, such as those which require multiple pages to be presented per experimental item (as in self-paced reading experiments),

and those that require flexibility in the way a page is displayed, are expected to be made possible in the future. However, there will always be some kinds of experiments that Speriment is not well-suited to. The goal is that in specializing by the needs of a field, Speriment will maximize both ease of use and power for a particular set of users. One strategy for those interested in using Speriment is to use Speriment for most experiments and to use psiTurk alone, which Speriment uses to interface with Mechanical Turk, for experiments that Speriment cannot express. psiTurk allows arbitrary JavaScript, removing all barriers to client-side development.

Although there is a tradeoff between ease of use and flexibility, certain techniques can be employed to increase one without losing out on the other. Speriment makes use of several such techniques.

First, Speriment separates the structure of an experiment from its content. Materials are read into a Speriment script from other files and inserted into Speriment components programmatically so that they don't have to be listed in the script. This makes experimental designs reusable and experimental stimuli easy to change, as well as making the scripts easier to read.

Second, Speriment strives for a declarative rather than procedural description of experiments. Procedural programming describes how a result is to be achieved, while declarative programming merely states what result is to be achieved. A procedural description of an experiment would issue commands to show the first page, then choose which page should be shown next, then accept a response, and so on. In contrast, the Python script describing a Speriment experiment builds structures and assigns them properties. The actions that will be taken over time are implicit in the structures and their properties and do not have to be described step by step.

Third, Speriment abstracts away from the details of commonly used experimental structures. The Latin square algorithm, used to distribute items and conditions across participants, provides a good example: the algorithm is fairly complex as compared to

other aspects of an experiment, as shown in its description in (1). Yet, it is commonly used among experimenters. Instead of giving experimenters the flexibility to build this algorithm as well as other less commonly used ones while burdening them with the responsibility for learning and implementing the algorithm, Speriment packages the concept of Latin Squares in a high-level feature that can be invoked with the code "`latin_square = True`."

(1)   Algorithm to produce a Latin Square

Begin with a list of groups 0 to n of pages 0 to m, and a version number v between 0 and m. For each group i, let the condition number c be $(i + v)$ mod m. Select from this group the cth page.

Frameworks like Turktools (Erlewine and Kotek, 2013) and psiTurk (McDonnell et al., 2012), discussed in more detail below, are associated with libraries of scripts that describe entire experimental designs. These scripts can be shared and reused across experiments and labs, but this job is only easy when the entire design is meant to stay the same. Libraries of Speriment scripts are also possible, but Speriment is built to make it easy to identify and reuse parts of an experiment, not just entire experiments.

With these properties of Speriment in mind, we can compare it to other frameworks commonly used to run web-based experiments in linguistics. In the following, I cover Ibex (Section 1.3.2); Experigen (Section 1.3.3); TurkTools (Section 1.3.4); Web-Exp (Section 1.3.5); the survey creation frameworks SurveyMan, SurveyMonkey, and LimeSurvey (Section 1.3.6); and the framework that Speriment relies upon, psiTurk (Section 1.3.7). Each package's documentation is written from a different point of view, making it difficult to compare packages feature for feature, so this comparison will focus on the highlights that may help experimenters decide which packages are best suited to their general needs.

1.3.2 Ibex

Ibex (Drummond, 2011) is a program that generates a website from a JavaScript description. Ibex provides JavaScript functions to describe the behavior of pages and the way they are ordered. Both of these aspects of Ibex are highly flexible; experimenters can write their own JavaScript implementing new patterns of page behavior, and the functions that block and order pages allow a wide variety of experimental structures. The limitation on its flexibility comes in at the level of interdependence between pages in an experiment. Ibex allows experiments to display feedback after a question that depends on the participant's response to the question, but other forms of interdependence, such as the conditional running of a block depending on a previous response, or some of the designs made possible by Speriment's `SampleFrom` feature (see Section 1.4.4), are not possible.

A drawback of Ibex format is that the stimuli are inserted into the code describing the experiment, making the design harder to read and the materials harder to change. A new version of Ibex is planned that will separate materials from design, as many other frameworks do, including Speriment (Alex Drummond, p.c.).

Ibex experiments can be run on IbexFarm, which is hosted by Drummond. Ibex-Farm eliminates the need for experimenters to run their own server. This is helpful, as running a server requires extra steps and incurs the risk that the server will go down and prevent participants from completing an experiment. A downside is that the server does not use a database to balance participants across conditions as psi-Turk's server-side code does. This means that Latin square and other designs that assign different participants to different treatments are possible, but usually require the experimenter to be involved in distributing the treatments to equal numbers of participants.

1.3.3   Experigen

Experigen (Becker and Levine, 2010) is a lightweight framework, likely to be less user-friendly for novice programmers but more flexible than many frameworks. Experimenters write the HTML to display their pages, allowing them to alter the layout of a page in ways Speriment does not currently allow. Similarly, experimenters write the design of their experiment in JavaScript, allowing them to build, select, and order pages however they like. This includes building dynamic pages, which vary across participants. Speriment can create dynamic pages using its `SampleFrom` component; in a typical example of the difference between the two frameworks, dynamic pages are created in Speriment by declaring them with `SampleFrom` and associated options, which provide a fixed repertoire of ways pages can be built, while Experigen simply allows experimenters to build pages however they like in JavaScript. Complicated patterns of dynamic pages will require more programming expertise to express in Experigen, but some patterns are possible in Experigen that Speriment simply doesn't provide.

1.3.4   TurkTools

Turktools (Erlewine and Kotek, 2013) is similar to Speriment in its specialization for a certain universe of experiments, which it makes very easy to implement, but different in its approach to making those experiments accessible. While Speriment does so by making their components easily expressible, Turktools achieves this with a library of "skeletons" for experimental designs commonly used in linguistics. While it is possible to write new skeletons, the process is not highly documented and would require programming expertise.

To use Turktools, the experimenter chooses a skeleton and creates a list of items, and runs the supplied Python scripts to create the experiment and to retrieve and

analyze the results. Turktools is associated with Turkserver, which can be used to run experiments without Mechanical Turk.

The display properties and behavior of each page are highly customizable, depending on the experimenter's familiarity with CSS and JavaScript. On the other hand, the Python scripts that process skeletons restrict the experimental designs that are possible. The possible designs include ones that are important in linguistics: Latin squares with instructions, randomized test questions, and demographics questions. Because the Python scripts are independent of each other, it would be feasible for a developer to extend the set of tools to fit a new type of experiment.

1.3.5 WebExp

WebExp (Keller et al., 2009b), like its successor WebExp2, does not seem to be currently supported, so it is included here for the feature comparison but not as a recommended alternative for running experiments. It is a framework in which the experimenter writes a description of an experiment in XML and a Java program interprets the description to run it on the web. Similarly, Speriment takes a description in JSON, a format similar in use to XML, and interprets it with JavaScript. Speriment, however, makes it easier to write this description by letting the experimenter write in Python and compiling the Python script into JSON.

WebExp provides a high degree of flexibility in the visual layout of the pages and in the timing of presentation of resources. It lacks support for audio resources, pseudorandomization, training blocks, and per-participant page construction. WebExp and Speriment have different ways of letting past performance in the experiment affect later pages: while Speriment allows past choices to choose among pre-determined pages, WebExp allows past responses to be inserted into later pages.

1.3.6 Survey Software

There are several websites and programs available for running online surveys, and some experiments are expressible using these tools. However, the difference in priorities for surveys and experiments means that many experiments are difficult or impossible to create using survey software.

Speriment grew out of a project to make a version of SurveyMan with the specific features needed to design experiments (Tosch and Berger, 2014). The two systems have different strengths. SurveyMan provides survey hosting, saving the experimenter the trouble of running their own server. SurveyMan randomizes question order by default, which is an unusual strength among survey software packages. SurveyMan is a particularly advanced tool for creating surveys because it enables the experimenter to detect which participants appear to have answered at random, so that their data can be excluded from the analysis. SurveyMan and Speriment share some features due to their common origin, such as exchangeable blocks, which allow some blocks of questions to be reordered across participants while others are guaranteed to maintain their position. However, SurveyMan lacks key experiment-specific features, such as the ability to automatically distribute questions according to a Latin square. The goal of Speriment is to address these needs.

SurveyMonkey (SurveyMonkey, Inc., 1999) is a popular tool for creating surveys, and has two advanced features: answer piping and skip logic. Answer piping inserts the answer to one question into a later question. Skip logic hides questions that are deemed irrelevant to the survey-taker based on their answer to a previous question. Speriment implements skip logic (referred to as conditional running of blocks, using the `RunIf` component), but not answer piping. However, SurveyMonkey lacks many key features for experiments. One of the most basic features for an experiment framework is randomization of question order, which is a premium feature in Survey-

Monkey. Other features, such as Latin squares and other schemes for reordering or choosing questions, do not appear to be available at all.

LimeSurvey (LimeSurvey Project Team / Carsten Schmitz, 2012) is an open source project primarily designed to enable the creation of online surveys. It has advanced features for survey designs that are sometimes applicable to experiments, as well, such as randomization of question order and conditional display of questions. Like WebExp, it can insert previous answers into later pages, and it can additionally insert participant-specific data, such as their email address, as default responses. However, it lacks some of the features that are often crucial to experimental designs, such as Latin squares, and does not appear to be able to insert data from an experimenter-supplied bank, as with Speriment's `SampleFrom` component.

1.3.7   psiTurk

psiTurk (McDonnell et al., 2012) is a framework for running experiments online that Speriment employs to handle the results database and integration with Mechanical Turk. It can also be used alone, in which case the experimenter must provide all client-side code in JavaScript. The makers of psiTurk are compiling a library of such JavaScript experiments to be shared among experimenters. Speriment makes it easier to create an experiment with psiTurk because the experimenter only needs to describe what the experiment *is* in Python, rather than describing what it *does* in JavaScript, as described in Sec. 1.3.1. Yet Speriment can only express a subset of the experimental designs one could implement in JavaScript.

1.4   Components and Features

A full user guide for Speriment is available on the GitHub repository at https://github.com/presleyp/Speriment, and example scripts are available on the repository as well as in Appendices A and C. However, it is useful to discuss the

components that make up a description of an experiment in Speriment and the features that each component currently provides.

1.4.1 Pages

Speriment descriptions are made up of pages. A page represents one view of the experiment; whatever information is displayed at one moment in time. These can be questions, instructions, or niceties such as a 'welcome' or 'thank you' page.

The order of pages is randomized by default. If they are assigned conditions, they may also be pseudorandomized so that no two pages of the same condition will appear in a row.

By default, all pages display eventually. However, the experimenter may create groups of pages when not all participants should see all pages. For a given participant, one page will be chosen from each group. This choice may be done randomly or according to a Latin square.

1.4.2 Options

Pages that pose questions should contain one or more options. Pages take arguments to describe properties of their option sets. Option sets can be exclusive, so that only one of them can be selected, or inclusive. Their order on the page is randomized, and they can be specified to be ordered or unordered. Ordered options, such as a scale of numbers, will not have their order fully randomized, but only kept in place or reversed. Finally, options can be free text, which produces a text box. Exclusive options display as radio buttons and inclusive options display as checkboxes, except if there are more than seven of them, in which case both exclusive and inclusive options are displayed in a drop-down menu. Options can be selected via mouse or keyboard.

1.4.3   Features of Pages and Options

Both pages and options can be associated with tags. Tags do not affect the implementation of the experiment, but serve to simplify the process of analyzing the resulting data. For instance, pages can be tagged with the type of the page, so that instructional pages can be easily filtered out. Options can be tagged with text relevant to the analysis. For instance, in an experiment designed to test whether people prefer cats or dogs, "Garfield" and "Stimpy" could have "cat" as the value for their "Species" tag while "Odie" and "Ren" have "dog." This eliminates the need to merge a table associating names with species into the table of results after running the experiment.

Pages and options can also have resources. Resources can be images, audio files, or video files. Page resources will display centered at the top of the page and option resources will display next to the option.

Speriment allows training blocks, which will be explained further below. Pages and options have two features to support the use of training blocks: correctness and feedback. Pages can specify which of their options is correct and options can specify whether they are correct or what regular expression a text answer must match to be correct. Pages can specify a feedback page that should show following the participant's response, or an option can specify a page that should show if it is selected. Correctness is used to determine whether the participant has mastered the task, and feedback can be used to help participants do so.

1.4.4   Sampling

The text, tags, correctness, and resources of pages and options are typically all specified with constants that remain the same across participants. However, they can also be specified via a `SampleFrom` component. `SampleFrom` takes the name of a bank containing text or the filenames of resources. These banks are passed to an

enclosing block or the entire experiment. The `SampleFrom` component will sample one string from the bank randomly at runtime, so that the selection varies across participants. By default, sampling is without replacement, so that once a string has been used for a page or option for a given participant it will not be used for a different page or option for that participant. However, there is an option for sampling with replacement. If a more complex relationship among items is needed, experimenters can specify a variable to associate with the sampled string. This variable can be any string or number. When the same variable is passed to two `SampleFrom` components, they will both sample the same string. Conversely, if the same string or number is passed as `variable` in one SampleFrom and `not_variable` in another, they will not sample the same string. To create relationships between different kinds of stimuli, the experimenter can put dictionaries rather than strings in banks. The `variable` and `not_variable` arguments manage the selection of a dictionary from the bank, and an additional argument, `field`, specifies which key to access in the dictionary to arrive at a sampled string.

The `SampleFrom` feature is one of the more complex and powerful features of Speriment. It enables experimenters to minimize correlations in their data without trying every possible combination of page components, such as text and images.

1.4.5 Blocks

Pages are grouped into blocks. While pages are shuffled by default, blocks display in the order in which they are specified by default. Thus, blocks enable experimenters to specify ordering. However, blocks can be made exchangeable. Exchangeable blocks are those that are allowed to switch places with each other. This feature is more powerful than one that allows shuffling, because it enables the first and third block, for instance, to trade places, while leaving a middle block in place. This can be useful for randomizing the order of test stimuli while leaving instructions and breaks in the

sensible spots. Experimenters can also mark blocks to be counterbalanced. Counterbalanced blocks behave in the same way as exchangeable ones, except that their location is deterministic rather than random. Counterbalancing is used in tandem with a psiTurk setting that categorizes participants; Speriment uses that categorization to choose an ordering for the counterbalanced blocks. Thus, counterbalancing is more assured to give a balanced distribution of orderings across participants, but depends on a categorization that can be used to determine other aspects of the experiment.

For instance, this categorization of participants by psiTurk also dictates the way treatments are assigned to participants. Blocks can also be assigned to treatments, and a given participant will only see blocks that are not in any treatment and blocks that are in their treatment, excluding blocks that are in a different treatment. Since treatments are determined through the same mechanism as counterbalancing, treatments should be paired with exchangeable blocks to avoid unwanted correlations when both features are desired.

Another way to decide if a block should run for a given participant is to base the decision on whether the participant answered a question a certain way. RunIf is an experimental component that specifies the condition that must be met for its enclosing block to run. It specifies a page and the answer that must have been given on that page for the current block to run — either the option that must have been selected or a regular expression that the text answer must match. Conditional running of blocks is useful in two types of situations. In the first, there is a choice between running a block and not running it or anything in its place. For instance, a question about whether the participant has studied linguistics could be followed up with a block of questions about linguistics classes if the answer was in the affirmative. The second case is a choice between multiple blocks. For instance, a question about

native language could be followed by a block of questions in whichever language was selected.

1.4.6 Planned Additions

Speriment is still being actively developed. One feature that will be a high priority is multi-page items. Currently, the page construct represents two independent concepts at the same time: a view, which is a webpage that is visible in one moment of the experiment, and an experimental item, which is a unit of data gathering that is independent from other such units to a certain degree. It is almost always best practice for items within a block to be presented in random order, but if an item requires multiple views, the views for a given item may need to be presented in fixed order. Currently, this can be approximated by wrapping the views for a given item inside of a block, but blocks are not meant to represent items, so they cannot undergo some processes useful for items, such as selection by Latin square. Thus, in order to implement self-paced reading studies and other designs that will be useful in linguistic research, a change is planned that will introduce an item construct separate of the page construct.

Another high-priority feature is to expand the options for the output format of the data. Currently, the data is written in a format that logs all relevant aspects of all options, but this creates a data file that is difficult to read into R and is best preprocessed with Python. A new feature is in the works to allow output data to be trimmed in advance on the assumption that the analysis in R will be based only on data from the option that was chosen (and that most experiments will rely on pages that only permit one option selection). This will allow experimenters to view all data in the full format in case they have questions about how the experiment was generated, and to analyze the data in R without writing an additional script.

These features will expand Speriment's abilities while maintaining its focus on being easy and useful for linguistic experiments. Further extensions will also be considered, and users can request or discuss features on the GitHub repository.

1.5   Workflow

Much of the workflow when using Speriment is the same as that for using psiTurk alone. The experimenter downloads and installs psiTurk, creates a new project, sets up a database, and edits the configuration file and templates provided by psiTurk. From there, the flowchart in Figure 1.1 illustrates how the experimenter interacts with Speriment and psiTurk. The diamonds represent files that the experimenter writes, and the parallelograms represent commands that the experimenter runs in their computer terminal. All other shapes represent files or programs provided or generated by Speriment and psiTurk. The rounded rectangle shows which part of the process is the typical psiTurk process for running an experiment.

The flowchart shows that the experimenter writes their materials (in a file named `my_materials.csv` for this example) and Python script (`my_experiment.py`). The Python script, described in more detail below, crucially imports Speriment's Python module and calls Speriment's `install` method. The experimenter then runs Python on the script (`python my_experiment.py`), which produces a JavaScript file describing the experiment (`my_experiment.js`). At this point, the workflow reconverges with the typical psiTurk workflow, as if the experimenter had written the JavaScript manually. psiTurk commands are run (`psiturk` and other commands, such as `hit create`) to set up and run the experiment, generating it as a website (*My Experiment*). psiTurk stores the data in a database which the experimenter can configure; if they do not, psiTurk will automatically create and configure a SQLite database. Finally, the experimenter can access their data from the database however they would
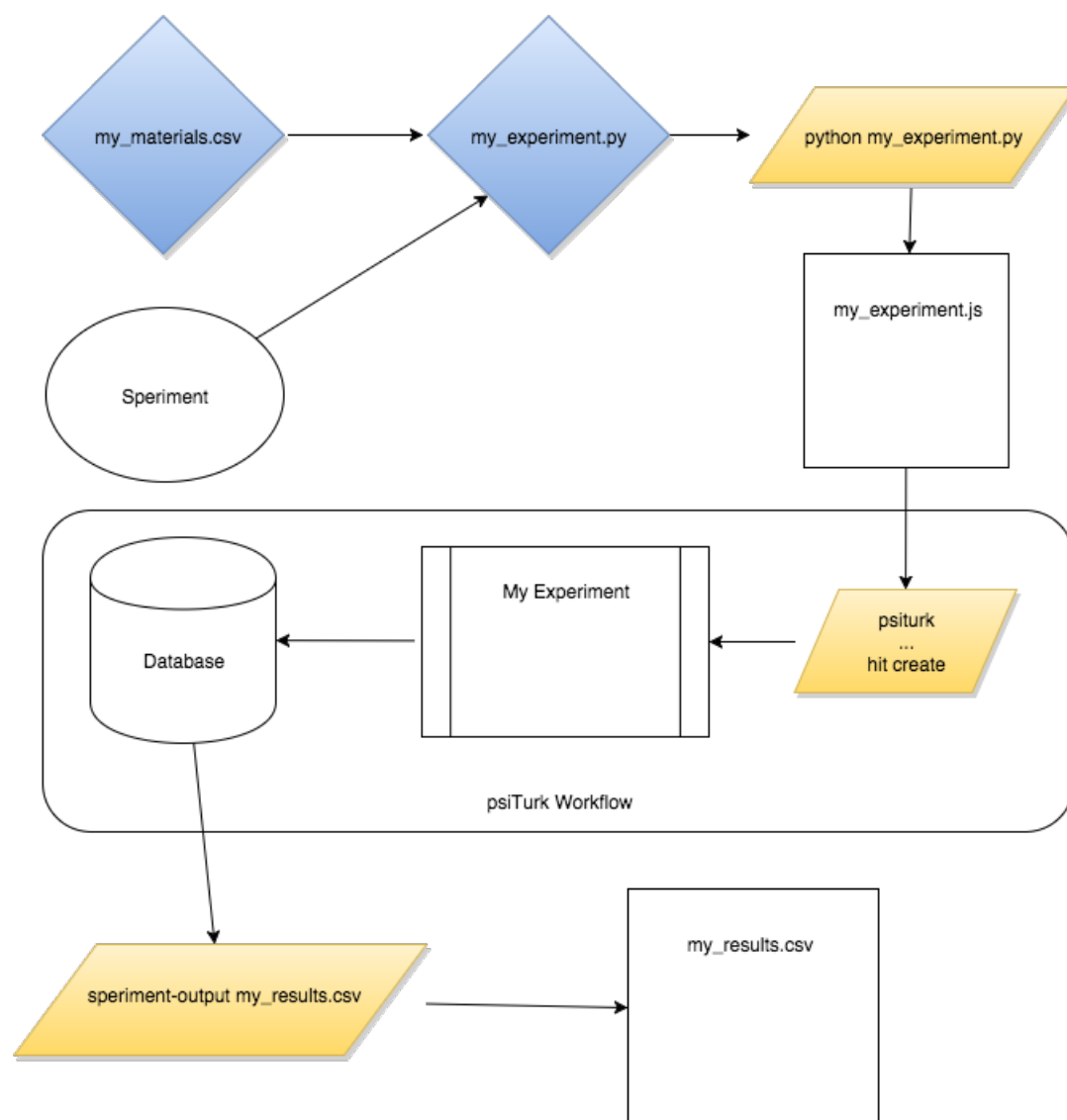
Figure 1.1. Speriment Workflow

like, but they are encouraged to make use of the `speriment-output` command to retrieve the data easily and put it into a JSON or csv file (`my_results.csv`).

A Python script used with Speriment usually contains the following parts:

1. An import statement, to make Speriment's Python classes and functions available in the script.

2. One or more lines, possibly using Speriment's `get_rows` or `get_dicts` utility functions, reading in data about the experimental stimuli. It is common to arrange the materials in a csv file with one row per item and one column per piece of information (text, condition, and so on) about the item.

3. A `with` statement creating an ID generator for use within the experimental components. This is an optional utility that can create unique identifiers automatically so that they don't have to be specified in the materials.

4. Lines of code creating pages and options. Frequently, this will be done with a for loop or list comprehension over the rows read in from a csv file. One page and one or more options will be created from each row. However, experimenters are free to create pages however they prefer; the materials do not need to be in any particular format in the csv file.

5. Lines of code creating blocks for the pages.

6. A line creating an experiment for all of the blocks.

7. A line naming and installing the experiment. This final line of the script validates the structure of the experiment, creates a JSON file describing the experiment, and edits psiTurk files to use both Speriment and the JSON file for this experiment. It is possible to keep more than one script in the project directory for different designs of an experiment. As long as they are assigned

different names, their JSON files will not interfere with each other. However, psiTurk only points to the experiment in the project directory that was most recently installed, so it's good practice to run the script right before launching the experiment.

These parts are labeled in the following example script:

```
[1] from speriment import *
[2] rows = get_dicts(my_materials.csv)
[3] with make_experiment(IDGenerator()):
[4]     my_pages = []
        for row in rows:
            my_pages.append(Page(row['text'],
                                 options = [
                                     Option(row['option1']),
                                     Option(row['option2'])])
[5]     intro = Block(pages = [Page('Welcome to my experiment!')])
        body = Block(pages = my_pages)
        goodbye = Block(pages = [Page('Goodbye!')])
[6] exp = Experiment([intro, body, goodbye])
[7] exp.install('my_experiment')
```

The scripts used to generate the experiments in this dissertation are given in Appendices A and C.

CHAPTER 2

CUMULATIVITY OF VIOLATIONS

2.1 Overview

In modeling constraint-based phonotactics, there are three broad kinds of decisions to be made: which framework to use (that is, which algorithms for learning from data and predicting new judgments), which parameters (constraints) to use, and how to tune (rank or weight) the parameters. This experiment will weigh in on the question of framework choice, using the function they use to combine the effects of violations to distinguish among them.

Several constraint-based frameworks for instantiating phonological grammars have been proposed. Optimality Theory (Prince and Smolensky, 2004) is a framework with strictly ranked constraints. This is in contrast with its predecessor Harmonic Grammar (Legendre et al., 1990; Smolensky and Legendre, 2006; Pater, 2009; Potts et al., 2010), which assigns weights to constraints. Linear Optimality Theory (Keller, 2006) is similar to Harmonic Grammar, but only allows constraints to assign penalties, not rewards; Harmonic Grammar, in contrast, can in principle allow both positively and negatively weighted constraints. Maximum Entropy Grammar (MaxEnt) (Goldwater and Johnson, 2003) is a version of Harmonic Grammar that converts the harmony scores given by Harmonic Grammar into probabilities. Hayes and Wilson (2008) have used MaxEnt to assign probabilities to all the surface forms of a language, rather than the surface forms of a given input candidate. I will henceforth refer to Linear Harmonic Grammar (Linear HG) to distinguish Harmonic Grammar without an exponentiation step from MaxEnt.

One feature of constraint-based frameworks is that they recognize units of violation that are independent of the words they appear in. In other words, two different words can be said to contain the same violation, and one word can be said to contain multiple different violations. Thus, these frameworks all have some method of combining units of violation into a grammaticality score for the entire word. However, they differ in how they define the combination operation. As a result, they make different predictions about the relative ungrammaticalities of words whose violations are in a subset-superset relationship. This experiment will seek to distinguish between frameworks on the basis of such groups of words.

In the following, I will use the term *penalty* to refer to the weight of a constraint multiplied by the number of times that constraint is violated.

2.2 Optimality Theory

Optimality Theory (OT) does not define an algorithm that computes a score for each candidate; it only defines an algorithm for choosing the winning candidate from a set of options. In order to evaluate OT's predictions for the cumulativity of violations, Albright (2008a) suggests that we interpret the grammaticality of a candidate as the maximum constraint penalty incurred by that word.[1] This interpretation predicts that adding mild violations to a word with a severe violation has no effect, so that the function from number of violations to grammaticality is flat for any given first violation as long as it remains one of the worst violations in the word. In other words, OT's function for combining the penalties of multiple violations is to return the maximum of those penalties as the penalty for the whole word.

---

[1]Constraint weights are not defined in OT, as constraints are only given relative ranks, so the term penalty must be interpreted loosely as the rank of the constraint accompanied by the number of violations of that constraint, rather than as the multiplication of the two. Nevertheless, the comparison can be made.

This is not the only possible way to extend OT to give scores to candidates. We could also imagine that when words are in a tie on their most severe violation, they are compared on their lesser violations, even if they are not in the same candidate set because they are both nonce words and thus not derived from the same input (Berent et al., 2001; Coetzee, 2004). This would not predict the above function, but it would still predict a lack of cumulativity of violations in some cases. OT does not predict gang effects, in which two mild constraint violations are together stronger than one severe violation (Pater, 2007).

Several experiments have shown, however, that both the potential and the definite failures of cumulativity in OT are not supported by speaker judgments. Predating OT, Ohala and Ohala (1986) found that speakers have an above chance probability of preferring a word with one violation to a word with that same violation and a less severe one, suggesting that even the milder violations affect the acceptability of a word. Assuming a straightforward transformation from grammaticality to acceptability, this contradicts the prediction that an additional violation that is lesser than the first violation will not affect the ungrammaticality of the word. Additionally, Coleman and Pierrehumbert (1997) found that a word like *mrupation*, with one severe violation followed by a common English sequence, was preferred to a word like *spleitisak*, with several minor violations. This is in contrast with OT's prediction that the strong violation *mr* matters more than any number of lesser violations. Keller (2000) found that in some cases, multiple violations of a constraint produce lower acceptability than a single violation in syntax, as well.

Albright (2008a) designed experiments to directly test the question of cumulativity of violations, addressing potential alternative explanations for these two results, and found that models that take into account all violations of a word, not just its worst violation, fit the data significantly better. Albright used two types of words, those with phonotactic violations in the onset and those with not only phonotactic violations

in the onset but also milder violations in the rime. In a variety of analyses, he fitted models that rate words by their worst violation only, and ones that rate words according to all their violations. The models that take into account all violations in the word were more strongly correlated with experimental findings. This study showed that cumulative models reflect speaker judgments better than noncumulative models, but did not distinguish among various cumulative models.

I conclude that OT's dearth of cumulativity effects is not empirically supported in the domain of phonotactics, and I turn to Linear HG and MaxEnt.

2.3   Linear vs. Exponential Combination

Albright (2008a) found evidence that all violations in a word contribute to the word's unacceptability, but he did not compare various models that work this way against each other. The shape of the curve relating number of violations to phonotactic judgments bears on the question of which framework we should use to model phonotactic well-formedness. Linear OT (Keller, 2006) and the nearly identical Linear HG have been used to model gradient well-formedness (Coetzee and Pater, 2008), so that, unlike in OT, there is a clearly defined way to determine the predicted cumulativity effects of these models. As Pater (2007) points out, Harmonic Grammar predicts a well-restricted set of cumulativity effects, unlike Optimality Theory with Local Constraint Conjunction (Smolensky, 2006). We can also find cumulativity predictions for the exponentiated and normalized version of HG, Maximum Entropy Grammar. Linear HG and MaxEnt predict differently shaped curves as violations accumulate.

(1)   Linear Harmonic Grammar: The harmony $\mathcal{H}$ of a word x is the dot product of the violation vector v, representing violations of x on each constraint in the constraint set C, with the constraint weight vector w.

$$\mathcal{H}(x) = \sum_{i \in C} v_i w_i$$

(2)   Maximum Entropy Grammar: The probability p of a word x is the exponentiated negative harmony of the word, normalized relative to the candidate set X.

$$p(x_i) = \frac{\exp(-\mathcal{H}(x_i))}{\sum_{j \in X} \exp(-\mathcal{H}(x_j))}$$

In order to illustrate the differences in how various constraint-based frameworks combine the effects of constraint violations, (3) gives four nonce words that contain zero to two constraint violations each, where the constraints are *mɹ and *osp. The former constraint is usually judged to lower word acceptability more and will be considered the stronger of the two, so I assign it a weight of 2 and *osp a weight of 1 for the purposes of this example. Example (3) shows the vector of penalties for each of the four nonce words.

Table 2.1. Example vectors of violations multiplied by constraint weights for nonce words.

Word	*mɹ	*osp
[ɹon]	0	0
[ɹosp]	0	-1
[mɹon]	-2	0
[mɹosp]	-2	-1

When Linear HG is applied to the example words and constraints given in 2.1, the candidates have the harmony scores 0, -1, -2, and -3; they decrease by two each time, in a linear pattern. In MaxEnt, if we assume these wordforms exhaust the possibilities, they have the probabilities 0.64, 0.24, 0.09, and 0.003; candidate A has the majority of the probability because it is the best choice available, and each additional violation decreases the probability by a smaller amount than the last.

Frisch et al. (2000) conducted several phonotactic judgment tasks and found that words predicted to be less acceptable by the model described in
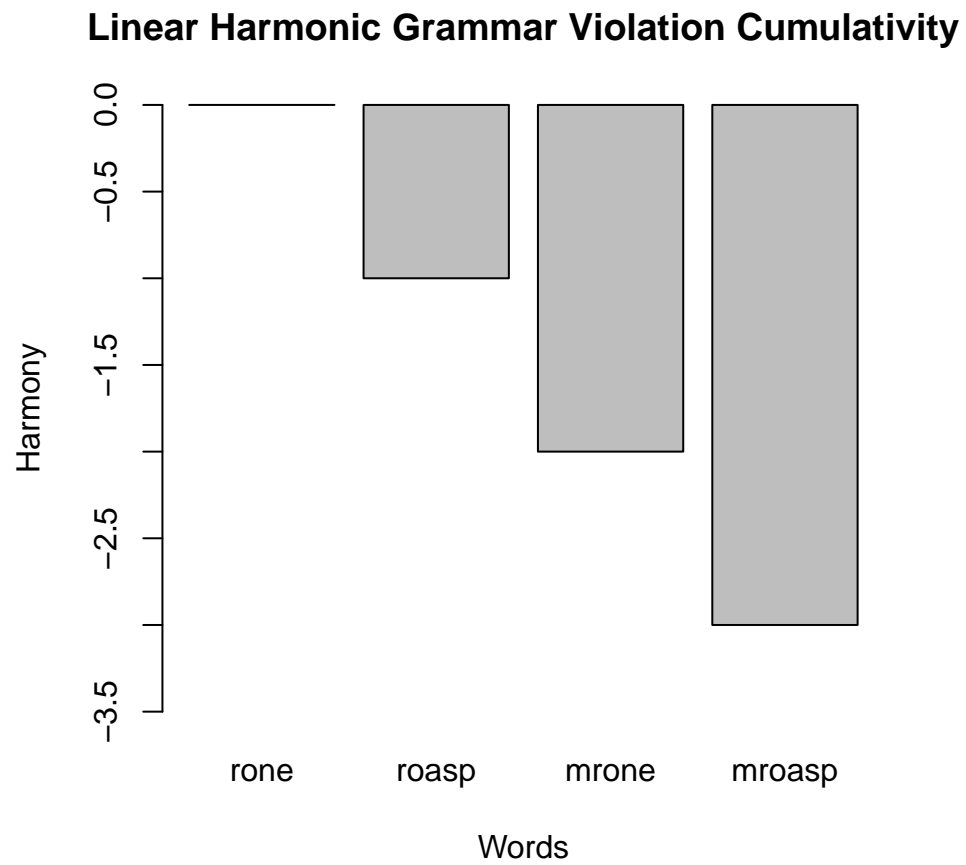
Figure 2.1. Cumulativity of violations in linear Harmonic Grammar.
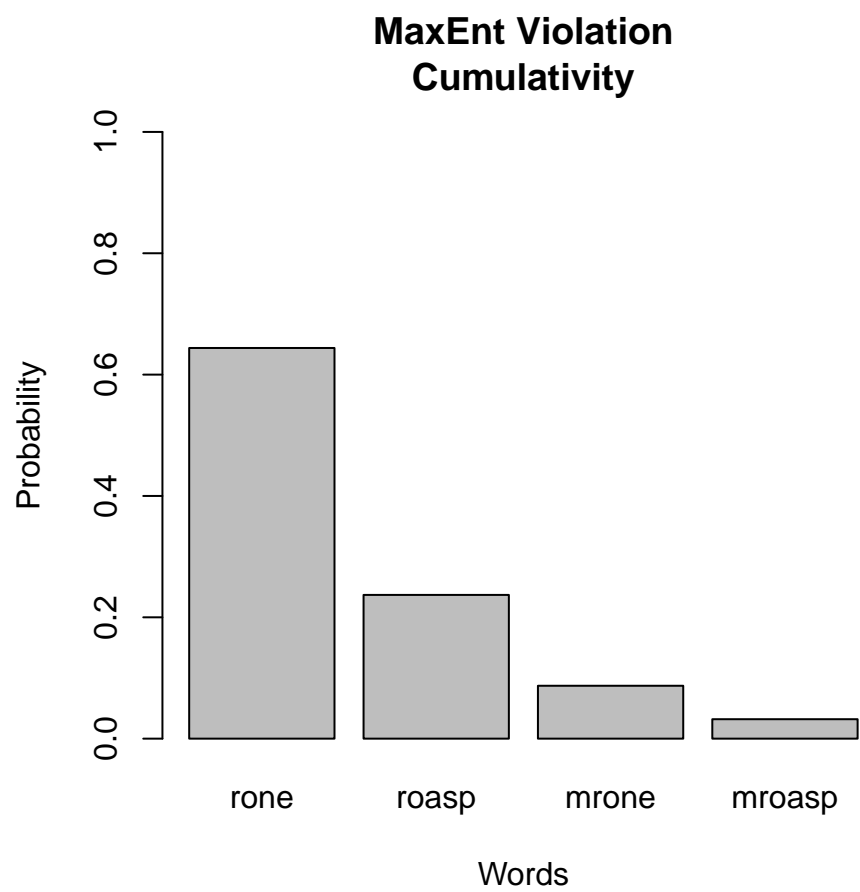
Figure 2.2. Cumulativity of violations in Maximum Entropy Grammar.

Coleman and Pierrehumbert (1997) were found to have less variability in their acceptability ratings than words predicted to be in higher parts of the acceptability scale. However, their experiments were not designed to test the accumulation of violations specifically; for instance, they did not control for the presence of one violation while adding another. Experiments 1 and 2 address this question directly by crossing two factors: the presence or absence of an onset violation and the presence or absence of a coda violation.

2.4   Experiment 1

This experiment uses evidence from the accumulation of violations to distinguish between Linear Harmonic Grammar and Maximum Entropy Grammar as models of phonotactic knowledge. I predict that, in accordance with the MaxEnt model, a violation in the presence of other violations will have a smaller effect on the acceptability of the word than it would have in isolation, as the acceptability approaches a floor.

2.4.1   Method

In order to test the prediction made by the MaxEnt model, I gathered acceptability data on words much like the ones used in the examples above: words with no obvious violations, words that are the same except with the addition of a violation in the onset, words the same as the first group except with a violation in the coda, and words with both the onset and coda violation.

2.4.1.1   Participants

One hundred participants were recruited from Mechanical Turk. They were each paid \$0.75, which is Massachusetts minimum wage for the amount of time the experiment was expected to last, five minutes. They were all located in the United States and claimed to be over 18 years old. In order to maintain a level of consistency in the participant pool, I ran the experiment only on weekdays between the hours of

noon and 5pm Eastern time, which corresponds to regular workday hours in the four continental US timezones.

Participants were excluded if they were not native speakers of English or if their data was suspect. Native status was determined by answers to two demographics questions: one asking their native language and one asking the language they use at home. Participants were only included if English was given in response to both of these questions. Other languages given in addition to English were not considered reason for exclusion. Two participants were excluded on the basis of native language.

The quality of the data was assessed in a variety of ways.

First, there were twelve filler trials. These had the same form as test trials, a yes/no question about the acceptability of a nonce word in English. However, six words were constructed to be more English-like than even the best test words, and six were constructed to be less English-like than even the worst test words. Participants were excluded if they accepted good fillers equally as often as bad fillers, indicating that they may not have been paying attention or answering carefully. Four participants were excluded for this reason. No participants accepted bad fillers more often than good fillers.

Second, any participant who consistently chose whichever option was on a particular side of the screen was excluded. "Consistently" was defined as more than 90% of test questions. No participants fell into this category.

The third exclusion criterion was answer speed. In pilot data, the fastest reaction times were over 300ms, so times under 50ms suggest that a computer program is clicking through the experiment automatically or that a participant is not reading the question. No participants were found to have this behavior.

After these exclusions, 94 participants were included in the analysis.

2.4.1.2 Materials

The words were presented orthographically, a practice sometimes used in phonotactic studies such as Daland et al. (2011). The benefit of visual presentation is that participants are less likely to fail to perceive the violations. There is a large body of evidence that speakers misperceive certain sound combinations that would be severe phonotactic violations in their native language. The result is that their behavioral data does not reflect the presence of the violation (Berent et al., 2007, 2009; Brown and Hildum, 1956; Dupoux et al., 1999; Hallé et al., 1998; Massaro and Cohen, 1983). If some violations were not perceived, the results of the experiment would be compromised, and indeed, some of the violations used in this study, such as [tl], are known to be among those misperceived by English speakers (Breen et al., 2013; Hallé and Best, 2007). It is possible that visual presentation will also cause the perception of illusory vowels, so the materials were constructed to avoid words that speakers are likely to understand as containing an unspelled vowel. For instance, *lb* was not included as a potential bad onset, because it was judged that speakers would be likely to interpret a word like *lbag* as having an intended pronunciation of [ləbag].

The downside of visual presentation as opposed to auditory presentation is that we may be testing orthotactics more directly than phonotactics. The materials were designed to minimize the ambiguity in the relationship between spelling and sound in order to mitigate this problem as much as possible.

Each of 24 test items appeared in four conditions. The four conditions are created by crossing two factors: presence of an onset violation and presence of a rime violation.

For each item, there was a unique good onset, bad onset, good coda, and bad coda. There was also a vowel assigned to the item. The four conditions of the item were created by taking all combinations of one onset, the vowel, and one coda.

The good onsets were all biconsonantal or triconsonantal and attested in native English words. The bad onsets were all biconsonantal, though some are represented

by three letters, such as *thl*. The bad onsets are not attested in native English words, although some are found in foreign proper nouns, such as *Sri Lanka* and *Vladimir*.

The good codas consisted of one consonant, sometimes spelled with two letters, such as *ss* or *ck*. The bad codas were biconsonantal. These codas are not attested in native English words, and are believed to violate English phonotactics. Many of them have sonority profiles not allowed in English, but extremely bad sonority profiles — those with a sonorant followed by an obstruent — were avoided to keep participants from mentally inserting a schwa into the coda cluster, breaking it into two syllables.

The vowels were taken from the set {*a, e, i, o, u, oo*}, in order to have a uniform distribution of orthographically distinct vowels across the items.

Items were created randomly from these components, and then altered to avoid any actual English words.

(3)   Test item components

    a.   Good onset: pl

    b.   Bad onset: tl

    c.   Good coda: g

    d.   Bad coda: vb

    e.   Vowel: a

(4)   Example test item

    a.   Good onset, good coda (GG): plag

    b.   Bad onset, good coda (BG): tlag

    c.   Good onset, bad coda (GB): plavb

    d.   Bad onset, bad coda (BB): tlavb

A Latin square design was applied to the test items so that each participant only saw one word from each item set.

Due to the Latin square, each participant saw six words from each condition. Accordingly, twelve fillers were added to the set of materials: six "good" fillers and six "bad" fillers. The good fillers were words expected to be even more English-like than the GG words, as they have real English suffixes added to them. Coleman and Pierrehumbert (1997) suggests that this addition will increase acceptability. The bad fillers were words expected to be even less English-like than the BB words, because they had longer violation-containing clusters. These fillers served two purposes: to reduce the chances of ceiling and floor effects in the test words by giving participants examples of more extreme acceptabilities, and to show when participants are answering randomly or inattentively. Attentive participants should accept good fillers more often than they accept bad fillers.

The full set of materials is provided in Appendix B.

2.4.1.3   Procedure

The experiment was built using Speriment and run using psiTurk (McDonnell et al., 2012) to interact with Mechanical Turk. The word components were put into a spreadsheet which generated the four conditions of each item. These items were read into a Python script, given in Appendix A, which described the structure of the experiment and supplied the items and their conditions to that structure. This script was compiled, and PsiTurk was used to post the resulting website on Mechanical Turk and manage interaction with participants.

The task is a two-alternative forced choice task between the responses "Yes" and "No" as answers to the question "Based on how it sounds, do you think this word could be a word of English?" followed by one of the stimuli. There were two options on the screen, one labeled "Yes" and the other labeled "No"; their ordering varied across items and participants.

Participants indicated their choice by pressing a key on their keyboard: 'f' for the choice on the left and 'j' for the choice on the right. The order in which the test and filler words were presented varied randomly across items and participants.

2.4.2 Results

As expected under any model, participants accepted the test item the most often when it had no violations, less often when it had one violation, and very rarely when it had two violations. Of the two types of words containing one violation, those with a coda violation and those with an onset violation, coda violation words were chosen less often, suggesting that the coda violations were on average more egregious than the onset violations. Thus, there is a total ordering of word types, from those with no violations (called GG for good onset, good coda), to those with an onset violation (BG for bad onset, good coda), to those with a coda violation (GB for good onset, bad coda), to those with two violations (BB for bad onset, bad coda).

2.4.2.1 Linking Hypothesis

The question this experiment seeks to answer is not about the ranking of the percentages for each condition, but the quantitative relationships among them. In order to analyze these relationships, we need a linking hypothesis to map from predictions about psychological states to predictions about performance on the task. I hypothesize that participants use the output of whichever model they are using — harmony in Linear HG and probability in MaxEnt — as the input to a probabilistic process that governs whether they choose 'Yes' or 'No' on the task. There are many forms this probabilistic process could take; I will adopt the assumption that the percent of times a participant accepts a word can be treated as a proxy for the output of the model for that word. That is, I assume that if participants calculate probabilities for words, they say 'Yes' to a word with the same probability they assign to the word, and if they calculate harmonies, they say 'Yes' with a probability that is proportional

to the harmony the assign the word. I assume that the scaling necessary to convert harmonies to probabilities in the case of Linear HG is constant across words, so that the differences in the percents of 'Yes' answers is the same, modulo the noise of the probabilistic process, as the differences in the harmonies. One way to achieve this is to translate the negative harmonies produced by Linear HG into a positive space, by simply adding the absolute value of the lowest harmony score to all harmony scores, and then normalize all resulting scores.

Thus, the linking hypotheses adopted here include a trivial one for MaxEnt, in which its output probabilities are interpreted without transformation as probabilities of acceptance, and a linear one for Linear HG, which does transform harmonies but in a way that preserves the ratios of harmonies.

The difference between the output of a model, harmony or probability, for GG words and the output of that model for another category of words can be viewed as the penalty for any violations — onset violations, coda violations, or both — that are present in the second category of words. In the case of BB words, which have both kinds of violations, this is a cumulative penalty, the combination of the penalty incurred by the onset violation and the penalty incurred by the coda violation.

Linear HG combined with the linking hypothesis described above predicts that the penalty for BB words is equal to the sum of the penalties for BG and GB, while MaxEnt predicts that the penalty for BB words is less than this sum.

Other linking hypotheses are of course possible; for instance, Legendre et al. (1990) proposed that the harmony of a sentence is transformed into its grammaticality via the logistic function, which pushes values towards the extremes of "grammatical" and "ungrammatical"; such a step is not assumed in this study.

Furthermore, the choice of a linking hypothesis is dependent on the experimental task. Many phonotactic studies use an acceptability scale, allowing participants to choose a level of acceptability for each test word. However, scalar data is difficult

to analyze, becasue different participants may use the scale differently. The distance between points on the scale would be crucial to investigating the shape of the curve relating violations to acceptability, and yet we cannot depend on all participants to assign the same distances between all points on the scale.

Another possible design is a two alternative forced choice task between two words rather than between two statuses of one word. In such a case, Luce's Choice Axiom (Luce, 1959) could provide a linking hypothesis from harmonies to probabilities of acceptance, in which the probability of accepting one word is equal to the harmony of that word divided by the sum of both harmonies. However, given a forced choice between a foil word of medium acceptability and each of the four types of test words, Linear HG and Luce's Choice Axiom predict a sublinear pattern of cumulativity not unlike that predicted by MaxEnt. This finding underscores the importance of making linking hypotheses clear and of future research into the validity of the hypothesis adopted here.

Noisy HG (Boersma and Pater, 2008) can also be used as a guide for how to convert harmony scores into probabilities of acceptance, since it models probabilistic data. However, the validity of this choice is debatable, as Noisy HG is intended to be a model of variation between surface forms that map from the same underlying form, not a model for comparing the phonotactic acceptability of unrelated forms. Furthermore, just as Linear HG paired with Luce's Choice Axiom can make a sublinear pattern of predictions, so can Noisy HG.

This design and linking hypothesis were thus adopted to make maximally distinct predictions for Linear HG and MaxEnt on the question of the linearity of cumulativity.

With our current limitations in understanding the transformations that apply to model outputs as they are used to direct behavior in an experimental task, the results of this experiment must be interpreted as dependent on the linking hypothesis assumed here. A more conclusive understanding of the phenomenon will depend on

future studies that investigate it using different tasks to determine if the findings are dependent on a particular linking hypothesis, or if they are robust to different ways of framing the question and to the particulars of different experimental tasks.

2.4.2.2 Analysis

As explained in the previous section, Linear HG and the linear linking hypothesis assumed here predict that BB words will be penalized as much as the sum of the penalties against BG and GB words, while MaxEnt and the trivial linking hypothesis adopted for it here predict that BB words will be penalized less than that amount. Table 2.2 shows that descriptively, the data appear to support the prediction of a MaxEnt model, as the penalties decrease at each step. Both report the percent of the time that participants responded 'Yes' to a word, for each type of word.

Table 2.2. Percent acceptance by condition in Experiment 1.

Condition	Mean Percent 'Yes'	Standard Deviation
GG	83	37.6
BG	30.9	46.2
GB	17	37.6
BB	6.6	24.8

The hypothesis that participants are employing MaxEnt rather than Linear HG predicts that there will be an interaction between the effects of onset violations and coda violations.

If a positive interaction is present, this would support a model that takes probability away from words in greater and greater amounts as the number of violations increases. There are known superadditive effects, where two violations incur a larger penalty than the sum of each independently (Albright, 2009a; Green and Davis, 2014), but this can be modeled with specific conjoined constraints rather than a consistently superadditive evaluation mechanism (Shih, 2015). It is not predicted that superad-

ditive effects are present consistently enough to affect the results of this experiment, which uses a wide variety of constraint violations.

A negative interaction would support models like MaxEnt, in which each additional violation subtracts less probability from the word than the last violation did.

A linear relationship would support models like Linear HG, in which each additional violation subtracts the same amount of probability from a word as the last violation did.

Figure 2.3 shows the interaction of the effects of coda violations and onset violations on 'Yes' responses. The difference between the slopes of the two lines shows that the addition of a coda violation decreases the acceptance rate less when an onset violation is already present than when it is not, as predicted by MaxEnt.

The interaction plot doesn't show the shape of the distributions, though, so we can also break down the data by participant and item in order to view violin plots of the conditions. A violin plot is a box plot where the sides of the boxes are replaced with kernel density plots. The white dots represent the median participant or item mean, the black bars represent the interquartile range — the values in the medial two quartiles of the data. Figure 2.4 is a violin plot where each data point is the percent of 'Yes' answers given to that word type by a particular participant. It shows us the distribution of participants for each condition. Figure 2.4.2.2 is a similar plot where the aggregation is done by item set. Recall that an item set is a set of four test words, one in each condition, where the words share substrings and constraint violations.

The violin plots reflect the same interaction as the interaction plot, supporting the trend found in the means with data from the medians and overall distributions. To test the significance of this interaction, a mixed effects model was fitted to the data using the lme4 package (Bates and Maechler, 2009) in R (R Development Core Team, 2011). The dependent variable was the participant's response to each word, coded as 0 for 'No' and 1 for 'Yes.' OnsetViolation, CodaViolation, and their interaction
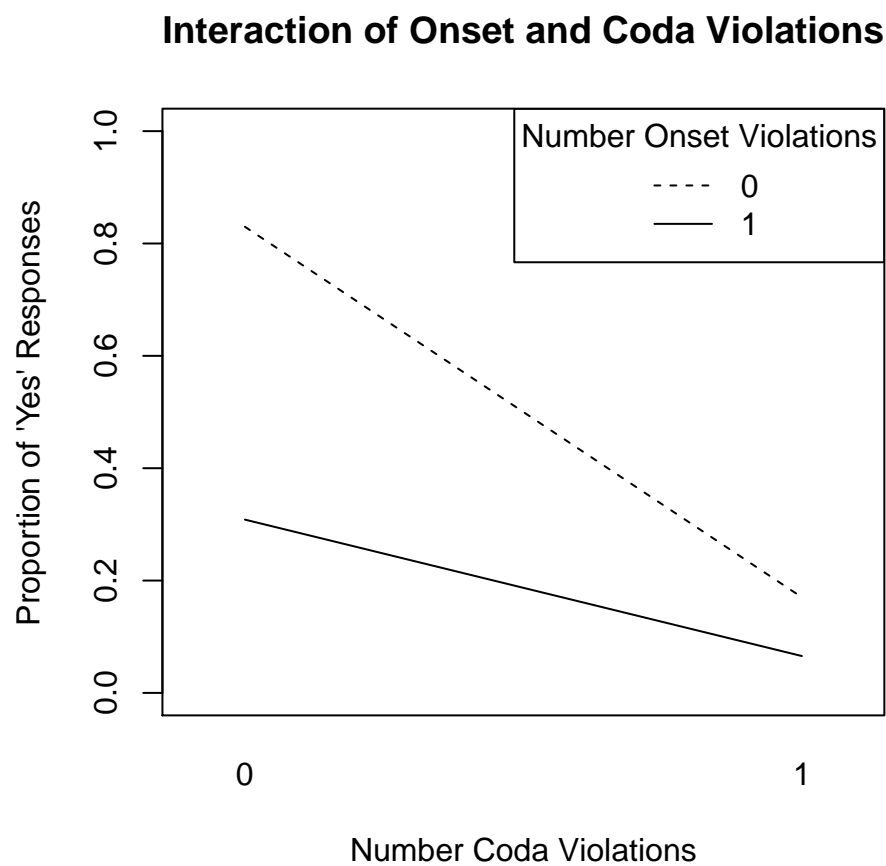
Figure 2.3. Interaction between onset and coda violations in predicting acceptance rate in Experiment 1.
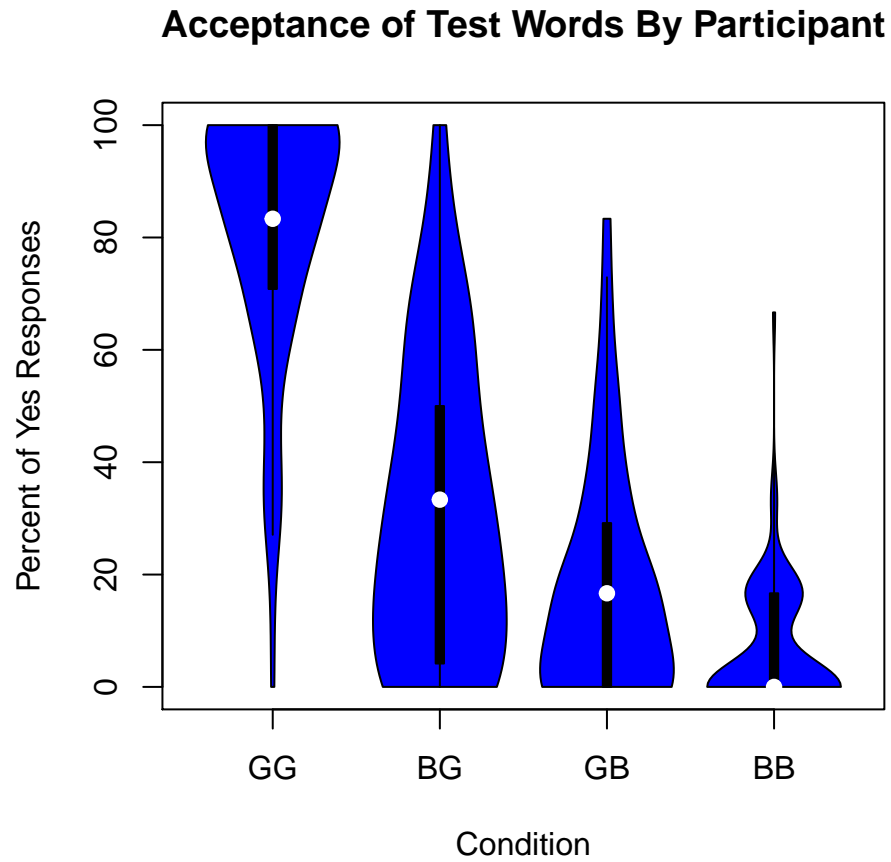
Figure 2.4. Distribution of mean percent acceptance by participant for each condition in Experiment 1.

Acceptance of Test Words By Item
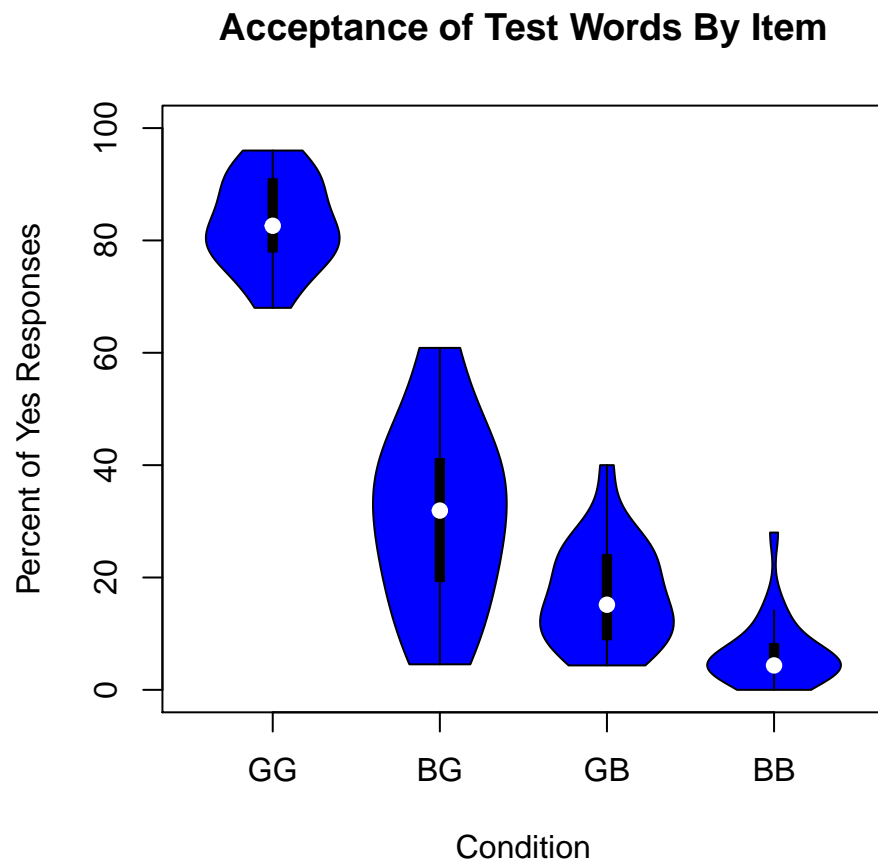
Figure 2.5. Distribution of mean percent acceptance by item for each condition in Experiment 1.

served as fixed effects. Random intercepts for participant and test item and random slopes for OnsetViolation, CodaViolation, and their interaction by participant and item were included in the model. Here and throughout the dissertation, the fixed and random effects structures were chosen by prioritizing a maximal random effects structure and over the inclusion of fixed effects that were not of theoretical interest. The maximal model that would converge within 20,000 iterations was used. The full model for the present analysis is given in (5).

(5)   Mixed effects model formula

Response $\sim$ OnsetViolation $*$ CodaViolation $+$ (1 | Participant) $+$ (0 + OnsetViolation | Participant) $+$ (0 + CodaViolation | Participant) $+$ (0 + OnsetViolation:CodaViolation | Participant) $+$ (1 | Item) $+$ (0 + OnsetViolation | Item) $+$ (0 + CodaViolation | Item) $+$ (0 + OnsetViolation:CodaViolation | Item)

Table 2.3 gives the coefficients of the mixed effects model. It shows significant effects of both OnsetViolation and CodaViolation, meaning that the presence of an onset violation significantly decreases acceptance rate, as does the presence of a coda violation. Additionally, it shows a subadditive interaction between the two; the main effects have negative estimates, because violations decrease acceptance rate, while the interaction has a positive estimate, because the coincidence of the two violations increased acceptance rate relative to the addition of their two main effects. This is in line with the MaxEnt prediction.[2] This interaction is significant at the $\alpha = 0.001$ level.

---

[2]The picture is complicated by the use of a logistic regression in this study, which means that the coefficients found by the analysis represent the effect the factors have on the log odds of acceptance rather than on the probability of acceptance. However, when transformed into probabilities, the probability of accepting BB words (0.097) is still higher than it would be if it were reached by adding the penalties against BG words (with a 0.233 probably of acceptance) and GB words (with a 0.101 probability of acceptance) relative to GG words (0.894 probability of acceptance).

Table 2.3. Coefficients of the mixed effects model in Experiment 1.

Factor	Estimate	p-value
Intercept	-0.87	4.4511041×10^{-6}
OnsetViolation	-0.84	7.8206064×10^{-8}
CodaViolation	-1.34	$4.2913714 \times 10^{-17}$
OnsetViolation:CodaViolation	0.82	4.8907995×10^{-7}

In Figure 2.4, we see that the distribution for BB appears cut off at the bottom. This suggests a floor effect: the BB words are so unacceptable that we cannot get an accurate sense of how unacceptable they are, because we already hit zero percent 'Yes' responses, and it's not possible to give a negative number of 'Yes' responses. This result is problematic for the interpretation of this experiment, which hinges on an accurate estimation of the acceptability of BB words. If participants are using an Linear HG-like grammar, perhaps it could map to a negative number of 'Yes' responses, which is then forced up to zero, making it appear that they employed a MaxEnt-like grammar instead.

One problem with this alternative interpretation is that it's unclear how harmonies would map to a negative number of intended 'Yes' responses. We would need a more sophisticated linking hypothesis than the one offered above, which assumed that harmonies are transformed into probabilities in a way that preserves their proportions. Implicit in that assumption was the idea that they were normalized to fit into the probability space and translated into positive space, keeping the proportions among them constant. Perhaps they are scaled but not fully pushed into the positive range, but it remains unclear what that would mean for our hypothesized probabilistic process to have a target of a negative number. The fact that harmonies must be shoehorned into a positive space in order to relate to real world behavior may be the very motivation for a MaxEnt-like model which transforms harmonies and results in the attentuation of differences at the bottom of the scale.

Furthermore, we might expect a more dramatic floor effect, and one consistent whether we look at the data aggregated by participant or by item, if indeed the participants intended to assign BB words a harmony score as low as Linear HG would predict for this data.

One clue to participants' true assessments of the BB words may lie in their reaction times. If BB words were far worse, not just slightly worse, than BG and GB words, and participants merely ran out of ways to express this, we might expect them to have shorter reaction times in responding to BB words than to BG and GB words, because it was so obvious that they should be rejected. In particular, we would expect shorter reaction times for rejecting BB words than for rejecting the second worst category, GB words. In fact, however, the violin plots for log-transformed reaction times for rejected GB and BB words look fairly similar, as shown in Figure 2.6.

Table 2.4. Central tendencies of log-transformed reaction times for rejecting GB and BB words in Experiment 1.

Condition	Mean	Median
Rejected GB	7.8901758	7.786126
Rejected BB	7.8504456	7.7454356

The mean and median reaction times for BB words are lower than those for GB words, as given in Table 2.4, but a one-tailed t-test finds that the difference is not significant, with a *t*-score of 0.975 and a *p*-value of 0.16501. Thus, reaction times do not offer support for the Linear HG hypothesis. However, we cannot accept the null hypothesis that BB and GB reaction times are the same. The evidence of a floor effect is inconclusive, motivating a second experiment to both replicate the interaction and differentiate it from a task effect.

2.4.3 Discussion

The results support the hypothesis that MaxEnt predicts participants' preferences better than Linear HG. The interaction between the effect of an onset violation and
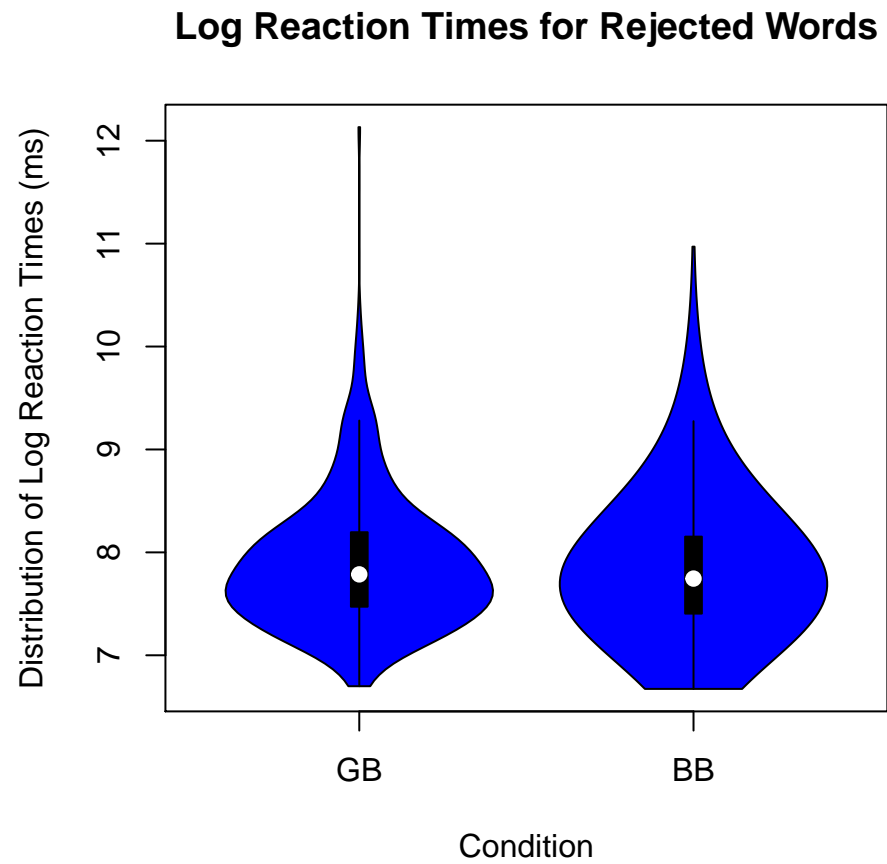
Figure 2.6. Distribution of log reaction times for rejecting GB and BB words in Experiment 1.

the effect of a coda violation was both statistically significant and of a considerable effect size. However, it is possible that this is a task effect, where the inability to accept BB words a negative number of times artificially inflated its measured acceptability, creating the subadditive interaction.

2.5 Experiment 2

Experiment 1 showed evidence of a floor effect, which interfered with the ability to interpret the results as conforming to the prediction of Harmonic Grammar or MaxEnt. Experiment 2 seeks further evidence to distinguish between these theories by using different materials to make a floor effect less likely and increase our ability to distinguish between a floor effect and an accurate measurement of acceptability. Experiment 2 is largely the same as Experiment 1, but with the materials altered to increase the distance in acceptability space between the violation-containing test words and the violation-containing fillers. This adjustment may encourage participants to distribute their responses differently, avoiding the low extreme of the acceptability space when assessing BB words, and it will also increase the likelihood that we can differentiate statistically between the BB words and an even less acceptable category of words, the bad fillers. If there is a category below the BB words, then they must not be at the "floor" of the task's measurable acceptability values.

2.5.1 Method

2.5.1.1 Participants

101 participants were run on Mechanical Turk, each paid \$0.75, as the experiment was expected to take the same amount of time as Experiment 1. Two participants were excluded for answering too quickly, implying they may be automated workers or not paying attention. Three participants were excluded for not being native or regular speakers of English. A total of 96 participants were included.

Participants were not excluded from this experiment on the basis of their responses to filler words, because the fillers serve a different purpose in this experiment and doing so could bias the results.

2.5.1.2 Materials

The materials for Experiment 2 have the same structure as those in Experiment 1 but differ in their exact makeup.

In order to construct these materials, the results of Experiment 1 were analyzed. The mean acceptance rates of each BG word and each GB word were calculated and sorted. The least preferred half of the bad onsets and the least preferred half of the bad codas were removed. In order to construct the same number of test words, the remaining bad onsets and bad codas were each used in two items. The good and bad onsets and codas were shuffled and recombined into new nonce words, which were filtered for real words or words with noticeable OCP violations. They are listed in Appendix B.

The good fillers were the same as in Experiment 1: nonce words with no known violations and real English suffixes. The bad fillers were made worse. They were extended to two syllables long, with violations in initial, medial, and final consonant clusters.

2.5.1.3 Procedure

The procedure in Experiment 2 was the same as in Experiment 1, except that an additional instructional page was used. This page was intended to anchor participants' expectations for acceptable and unacceptable words by giving them an example of a word they would choose "Yes" for (a word constructed in the same manner as the good fillers) and an example of a word they would choose "No" for (a word constructed in the same manner as the bad fillers).

The reworking of the materials was done outside of Speriment, as was their initial construction. Speriment was then used to implement the experiment and PsiTurk was used to run it on Mechanical Turk, as in Experiment 1. The code is similar to that of Experiment 1 and can be found in Appendix A.

2.5.2 Results

As in Experiment 1, grand means show that the GG words were accepted the most often out of the test words, followed by BG, GB, and then BB, indicating that more violations decrease acceptability and that the coda violations were on average considered worse than the onset violations. Table 2.5 gives the mean and standard deviation of the percent of accepted words from each condition, including the good and bad fillers. Unlike in Experiment 1, the bad fillers will be used in part of the analysis, so the filler acceptance rates are shown throughout the results of this experiment.

Table 2.5. Percent acceptance in Experiment 2 by condition.

Condition	Mean Percent 'Yes'	Standard Deviation
Good Filler	91.7	27.7
GG	88.7	31.7
BG	46.9	49.9
GB	29	45.4
BB	14.6	35.3
Bad Filler	6.8	25.1

Figure 2.7 shows an interaction plot of the data. As before, the slopes of the lines are different, suggesting an interaction. The acceptability of a word with an onset violation falls less with to the addition of a coda violation than the acceptability of a word without an onset violation does. All mean acceptance rates are higher in Experiment 2 than they were in Experiment 1, due to the change in materials.

Figure 2.8 shows a violin plot of the mean acceptance rates of each participant on each of the six conditions. The pattern among the test words is similar to that found in Experiment 1, with the largest difference in adjacent conditions occurring between
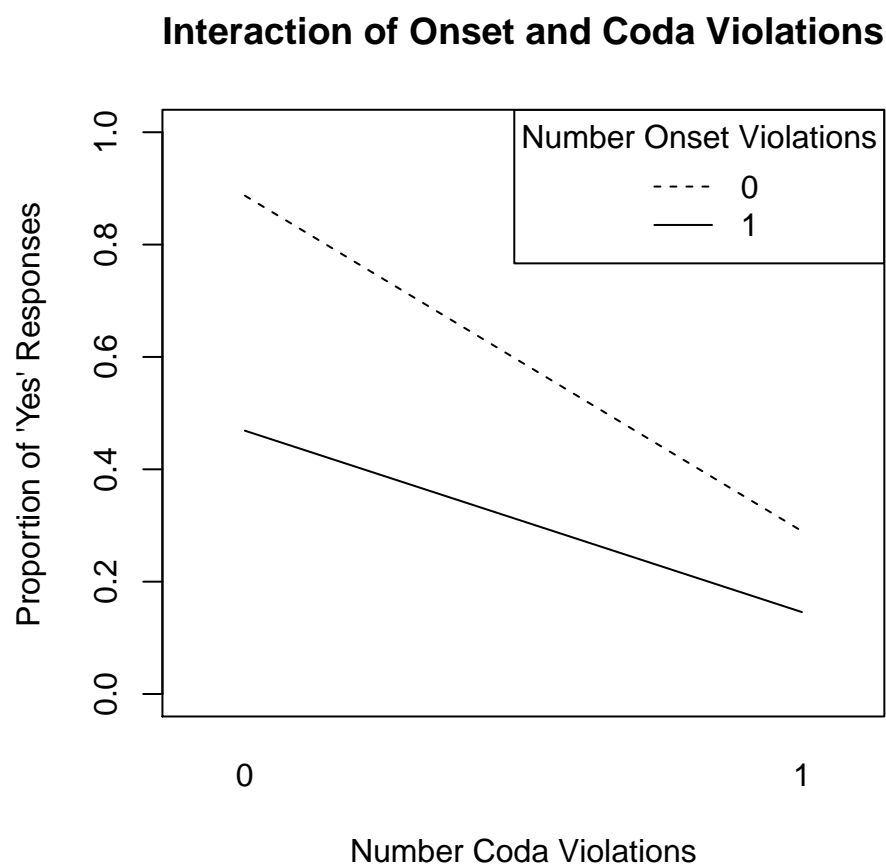
Figure 2.7. Interaction between onset violations and coda violations in Experiment 2.

the GG and BG words. The GG and BB words do show a possible ceiling and floor effect, respectively. The distribution of the good fillers is similar to that of the GG words, but with a smaller variance; the same is true of the bad fillers relative to the BB words.

Acceptance of Test Words by Participant
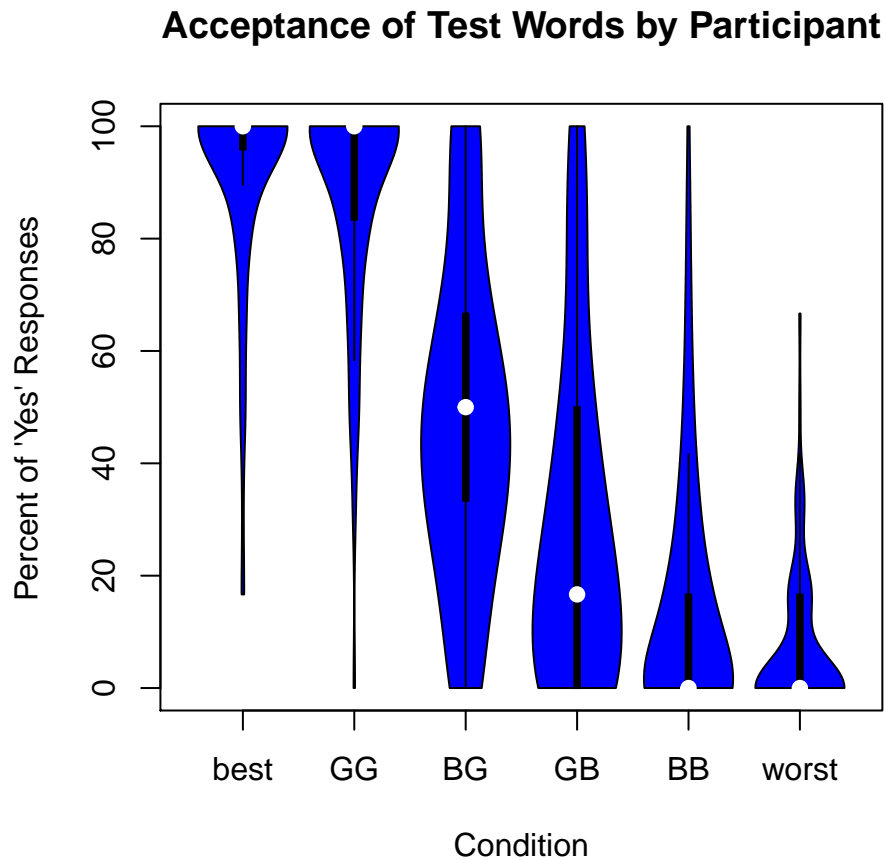
Figure 2.8. Distribution of mean percent acceptance by participant for each condition in Experiment 2.

Figure 2.5.2 shows the data aggregated by item instead of by participant. As in Experiment 1, it appears that participant means vary more overall than item means do. The by-item data shows less evidence for a floor effect, as the bad fillers have a noticeably lower distribution of mean acceptance rates than the BB words do.

Acceptance of Test Words by Item
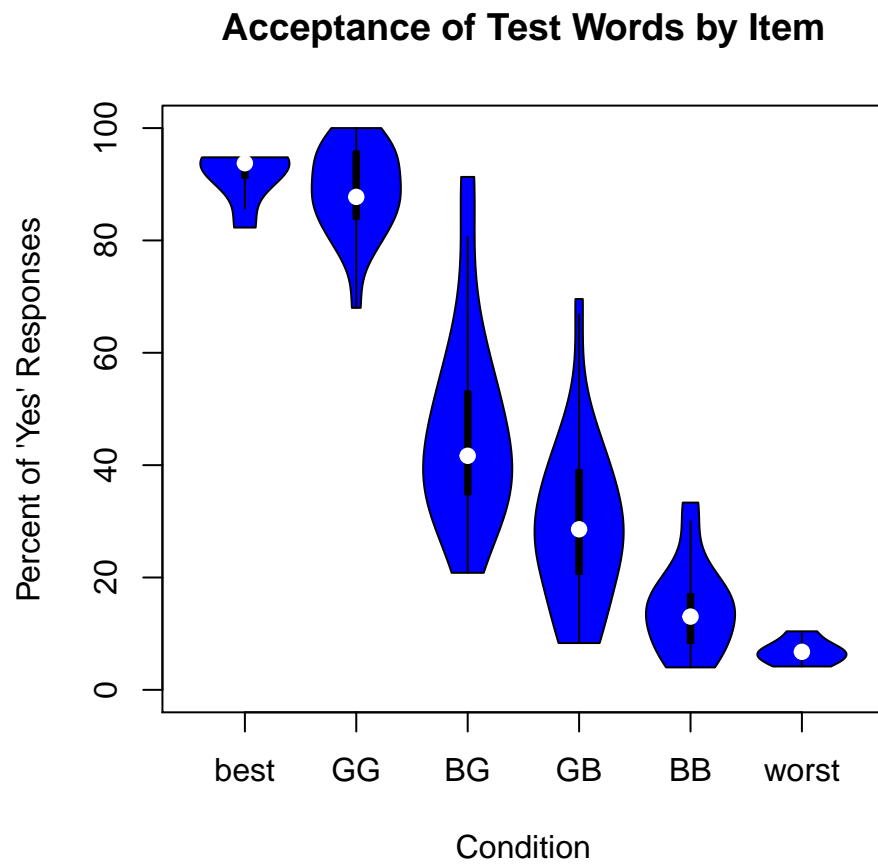
Figure 2.9. Distribution of mean percent acceptance by item for each condition in Experiment 2.

As in Experiment 1, a mixed effects model was fitted to the data. The model included fixed effects of theoretical interest and a full random effects structure for those fixed effects, and showed significant effects of OnsetViolation, CodaViolation, and their interaction. The formula is given in (6). The coefficients of the model are given in Table 2.6.

(6)   Mixed effects model formula

Response $\sim$ OnsetViolation $*$ CodaViolation $+$ (1 | Participant) $+$ (0 $+$ OnsetViolation | Participant) $+$ (0 $+$ CodaViolation | Participant) $+$ (0 $+$ OnsetViolation:CodaViolation | Participant) $+$ (1 | Item) $+$ (0 $+$ OnsetViolation | Item) $+$ (0 $+$ CodaViolation | Item) $+$ (0 $+$ OnsetViolation:CodaViolation | Item)

Table 2.6. Coefficients of the mixed effects model in Experiment 2.

Factor	Estimate	p-value
Intercept	-0.29	0.0944687
OnsetViolation	-1.07	1.230616×10^{-22}
CodaViolation	-1.68	$2.0908335 \times 10^{-35}$
OnsetViolation:CodaViolation	0.46	4.4394417×10^{-5}

The main effects were significant, showing that constraint violations lower acceptability. Their interaction was also significant at the $\alpha = 0.001$ level. As predicted by MaxEnt, and in agreement with Experiment 1, the interaction is subadditive, having a coefficient of opposite sign of the main effects.

In order to assess whether the BB condition was subject to a floor effect, meaning that the interaction is evidence of a task effect rather than evidence of the way the phonotactic grammar computes acceptability, a t-test was performed on the BB words and the bad fillers. The analysis was done this way instead of in the regression because the fillers are not captured by the crossed factors used in the analysis. The prediction for the t-test is based on the fact that the bad fillers contain multiple onset and coda

violations each, rendering them worse than the BB words. Therefore, if the task is capable of showing a statistically significant difference between the bad fillers and the BB words, then we can conclude that the performance on the BB words is not due to a floor effect. Indeed, a one-tailed paired t-test gives a t-score of 4.463 and a p-value of 4.8618589×10^{-6}. In order to control for Type I error, we should apply a Bonferroni correction to the results of the mixed effects model and the t-test, adjusting the α level to $\frac{0.05}{2} = 0.025$. Both the p-value of the interaction and that of the t-test are below this new threshold, so we can conclude that both are statistically significant.

2.5.3 Discussion

This experiment finds support for the hypothesis that a violation lowers acceptability less in the presence of other violations than it does in isolation. Not only was this pattern observed and found to be significant, but it was also found to be distinguishable from a task effect. The concern that the condition containing multiple violations was assigned artificially high acceptability due to the nature of the task is undermined by the existence of a further condition assigned lower acceptability under the same task. The findings are consistent with MaxEnt models but not with Linear HG models.

2.6 General Discussion

Experiments 1 and 2 reach the same conclusion: there is evidence that a violation in the presence of another violation contributes a smaller penalty to word acceptability than that violation does in isolation. This conclusion is consistent with a MaxEnt model and challenges a Linear HG model.

These results also offer further support to the conclusion reached by Albright (2008a) that an Optimality Theory style of violation combination in which violations of different constraints do not accumulate is not correct for phonotactic judgments.

The BB words may not be as bad as a linear model would predict, but they also do not seem to be as good as the GB words, indicating that the more mild onset violation still affects the word's overall acceptability.

The results of this study cannot, of course, single out MaxEnt as the correct grammar. There may be other models that would also fit the data gathered here. Furthermore, MaxEnt makes more specific predictions than that of subadditive cumulativity of violations, and further work is needed to test whether those predictions are also supported by the data.

However, these findings do pose a challenge to Linear HG as a model of cumulative phonotactics and support the idea that violations make more of a difference at the high end of the acceptability scale than at the low end. This idea can bear on the question of whether grammaticality is categorical or gradient, which Hayes and Wilson (2008) point out has been a challenging question for decades. If differences matter more in one region of the scale than another, this can make elicited intuitions appear to show a threshold between grammatical and ungrammatical data, offering a sort of reconciliation between categorical and gradient views of phonotactic well-formedness.

Another insight this experiment can bring has to do with the very high region of the acceptability scale, which is often overlooked. The good fillers in Experiment 2, made from violation-free words with English suffixes, were rated slightly higher than the GG (violation-free) words. This raises challenges for a theory of grammar that uses only violations and no rewards, such as Keller (2006). It is suggestive of analogical approaches to grammar, which are straightforwardly capable of rewarding forms that resemble existing words. However, it is possible that a grammar abstracted from the lexicon, such as constraint-based grammars, could also account for this phenomenon by incorporating constraints that reward the use of existing affixes in addition to penalizing the use of marked sequences.

58

The fact that the finding was replicated with a different set of materials can increase our confidence in the conclusion that violations accumulate nonlinearly. However, further variations on this experiment would be helpful. Auditory stimuli would help ensure that we are measuring phonology rather than orthographical effects, although it would be important to find stimuli that can be accurately perceived. Testing speakers of other languages on violations of their phonological constraints would further test the robustness of the effect, and show whether it reflects something about human grammar rather than a fact particular to English or the kinds of constraints available for testing on English speakers. Investigations of the cumulativity of constraint violations that rely on different tasks and designs would be helpful in reducing our reliance on the linking hypothesis adopted for this experiment.

CHAPTER 3

EFFECT OF ALTERNATIONS ON PHONOTACTICS

3.1 Background

Phonological generalizations can be divided into two types: phonotactic general-
izations and alternations. Phonotactic generalizations are true of the words in the
lexicon when they are treated as meaningless strings with no relationships to each
other. For instance, one could say that a language has no word-final voiced obstru-
ents, or no falling sonority onsets. Alternations, on the other hand, require reference
to the relationships between words. When two words share a morpheme, and that
morpheme takes a different phonological form in one word than in the other, we
say that the morpheme alternates. For instance, the English plural suffix alternates
among the forms [s], [z], and [ɨz]. Phonotactics are often referred to as static general-
izations because they are true of forms without any transformations needing to occur,
whereas alternations are referred to as processes, implemented via rules, under the
view that underlying forms change via these processes into their surface forms, and
the application of different processes or the application versus the non-application of
a process produces the different alternants of the morpheme.

It has long been recognized that the phonotactics and alternations of a language
share some generalizations in common. For instance, Russian words lack obstruent
sequences that disagree in voicing, and also has voicing assimilation alternations to
prevent the creation of such sequences (Jakobson, 1978). Chomsky and Halle (1968)
advocated against redundantly representing these generalizations in favor of encoding
them as rules that apply both across forms of a morpheme and "internally to a lexical

60

item" (p. 382). Similarly, Kisseberth notes that morpheme structure conditions, that is, phonotactic generalizations, often effect the same result as some of the rules in a language (Kisseberth, 1970:294). Furthermore, Kisseberth showed that multiple rules can participate in a "conspiracy" in which they all result in maintaining the same existing phonotactic generalization in the language. He described a conspiracy in Yawelmani, in which four generalizations participated in a conspiracy to avoid triconsonantal clusters: no morphemes contain them underlyingly, a vowel epenthesis rule avoids them, and two consonant deletion rules avoid them. In order to fully capture the insight that all of these generalizations are driven by one force, such as a constraint *CCC or *ComplexCoda, we must assume that phonotactics and alternations are encoded in the same grammar and share such a constraint. Otherwise, we can trace the three alternations back to one source, but we can't unify the cause of the alternations with the cause of the phonotactic generalization. Optimality Theory (Prince and Smolensky, 2004) formalizes this insight by using one set of constraints to model both phonotactics and alternations.

However, there are also cases where phonotactics and alternations do not work towards the same end results, but instead alternations avoid surface forms that are phonotactically acceptable. For instance, Pierrehumbert (2006) shows that the velar softening rule in English generalizes to novel Latinate-sounding words, such as [klɛmɪk] ∼ [kləmɪsɨɾi]. In addition to her arguments that velar softening is not phonetically grounded, she shows that it does not generalize to Germanic-sounding words such as [blɛk] ∼ [blɛkɨɾi], indicating that the alternations in the Latinate words is not necessarily required by the phonotactics. This can be interpreted as showing that alternations behave differently from phonotactics, or motivate more complex representations of the phonotactics of English.

Another phenonemon that demonstrates some independence between alternations and phonotactics is derived environment effects. First discussed in generative phonol-

ogy by Kiparsky (1973); Mascaró (1976) and reviewed in Wolf (2008), derived environment effects occur when a process applies only in derived contexts, that is, contexts in which some other process has applied or an affix has been added. Thus, the phonotactic system tolerates sequences in underived environments that alternations eliminate in derived environments. These effects have been analyzed in OT-based theories, however (Lubowicz, 2002; McCarthy, 2003; Van Oostendorp, 2007; Wolf, 2008; Anttila, 2009); depending on the types of constraints allowed and the relationships between inputs and outputs, it is possible to handle this apparent divergence between phonotactics and alternations in a system that derives both from one constraint set.

The reverse situation is also observed, where monomorphemic words do not have a particular sequence, indicating a phonotactic ban, but that sequence is permitted at morpheme boundaries. Martin (2011) gives two examples: Navajo sibilant harmony is respected in roots but can be violated in compounds containing multiple roots (Sapir and Hoijer, 1967; Young and Morgan, 1980), and geminates are banned within morphemes in English but can occur across morpheme boundaries (Hammond, 1999). Depending on theoretical assumptions, these multimorphemic words may also be subject to the phonotactic system, but they are a place where we might expect alternations to bring surface forms into line with the phonotactic generalizations that are true of single morphemes, as they so often do. Martin shows, however, that these languages exhibit a tendency to respect the phonological constraints across morpheme boundaries, even though there are exceptions, and demonstrates that a Maximum Entropy learner can account for this loose coupling between tautomorphemic and heteromorphemic constraints.

Thus, it is difficult to draw conclusions from the typology alone about the degree of interaction between the two types of knowledge.

The evidence from acquisition offers support, albeit weak, for the view that phonotactics and alternations are independent. At nine months of age, infants show sensitiv-

ity to the phonotactic generalizations of their native language. They can distinguish native words from foreign words (Jusczyk et al., 1993), phonotactically acceptable nonwords from phonotactically unacceptable nonwords (Friederici and Wessels, 1993), and words that are more probable in their language from words that are less probable in their language (Jusczyk and Luce, 1994). However, at this stage they do not seem to be able to acquire alternations. White et al. (2008) show that at 8.5 months of age, infants distinguish sequences in an artificial language based on their transitional probabilities, but do not treat alternants of a morpheme as "the same" morpheme when transitional probabilities are not available as a cue. In contrast, they found that 12 month old infants do treat alternants of the same morpheme differently from unrelated morphemes, suggesting in this time window, infants develop the ability to learn alternations.

That alternations are acquired later than phonotactics is somewhat to be expected, as phonotactic knowledge can be learned from the speech stream, while alternations depend on lexical knowledge (Hayes, 2004; Adriaans and Kager, 2010). It is possible that once alternations are learned, they are encoded in the same grammar as the phonotactics. However, we might expect that if the two share a grammar, infants would behave as if they know alternations that are motivated by a phonotactic generalization that they have learned.

Pater and Tessier (2003) investigated the relationship between phonotactics and alternations in adults, by asking whether knowledge of a phonotactic generalization affected the ease with which an alternation is learned. If the two kinds of knowledge are held in the same system, then an alternation motivated by a known phonotactic generalization should be easier to learn than one motivated by a previously unsupported phonotactic generalization. They found that this was the case, offering support for the view that phonotactics and alternations are either encoded in the same grammar or have a pathway for sharing information. They note, however, that another

interpretation of their results is possible: the alternation not supported by English phonotactics is not only novel to English, but phonetically unmotivated. If speakers are biased towards learning phonetically grounded generalizations (see for instance Becker et al. (2011), c.f. Hayes et al. (2009)), this alternation may have been harder to learn for that reason rather than because it lacked support from the phonotactic system.

Even if their result is due to communication between the systems encoding phonotactics and alternations, their experiment investigated only one direction of communication: from phonotactics to alternations. Thus, the question of whether alternation-based knowledge affects phonotactics remains open.

This question is relevant to the way work in phonotactic modeling is carried out. Often, phonotactic models are built to ignore morphemic and lexical information, learning phonotactic generalizations in the absence of alternations (Coleman and Pierrehumbert, 1997; Bailey and Hahn, 2001; Vitevitch and Luce, 2004; Albright, 2009b). This has also been true of some constraint-based models in the Optimality Theory tradition. Adriaans and Kager (2010) model the acquisition of phonotactics from the unsegmented speech stream, as a proof of concept of how infants may learn phonotactics and use it to help them learn words. In this case, learning alternations would be beside the point, as the goal is to demonstrate the learning of phonotactics when lexical information is not available. Hayes and Wilson (2008) model the induction of phonotactic constraints from words, but these words are not associated with meanings or broken into morphemes, so alternations are not represented. As this learner is used to model adult data, and has been used to assess the learnability of phonotactic generalizations given the lexicon of a language (Daland et al., 2011), it is important to know whether its omission of the influence of alternations on constraint identities and weights is an accurate representation of reality or a simplification.

There are also constraint-based models of learning in the Optimality Theory family that assume that alternations and phonotactics do interact (Tesar and Prince, 2004; Jarosz, 2006). In this case, it is also important to know whether the flow of information the models depend on is actually possible and should be appealed to in order to explain empirical findings.

If no evidence is found for the effect of alternation-based data on phonotactic knowledge, researchers modeling phonotactics would be justified in abstracting away from morphological data, simplifying the modeling process. If, on the other hand, such evidence is found, it would motivate new work in phonotactic modeling and underscore the utility of theories that capture conspiracies across the two domains. Thus, it is of considerable interest whether information from alternation data affects the phonotactic grammar.

3.2 Experiment 3

In order to address this gap in our understanding of the interaction of parts of the phonological grammar, I conducted an experiment to test whether learning an alternation affected participants' phonotactic judgments on underived words. It is difficult to find a case in natural language with the properties necessary to do a well-controlled test of this nature, because the phonotactic evidence for the generalization must be controlled in order to test the effect of the alternation. Thus, I use an artificial language learning paradigm.

3.2.1 Method

In order to test whether alternations affect phonotactics, this experiment manipulated alternation-based evidence while keeping phonotactic evidence constant.

Two constraints were constructed: one against a voiced obstruent followed by a voiceless obstruent, and one against a nasal followed by an obstruent of a different

place. One rule was constructed to repair each constraint: disagreement in voicing is repaired by devoicing the first obstruent, and place disagreement is repaired by changing the place of the nasal.

The formal definitions of the constraints are given below along with short names for them. These names are not conventional but match their use in the artificial language, to make the experimental design easier to follow. As will be shown below, the second segment in any constraint violation in the artificial language is always [f].

(1) *DF

$$* \begin{bmatrix} +\text{voice} \\ -\text{sonorant} \end{bmatrix} \begin{bmatrix} -\text{voice} \\ -\text{sonorant} \end{bmatrix}$$

(2) *NF

$$* \begin{bmatrix} \alpha\text{place} \\ -\text{continuant} \\ +\text{sonorant} \end{bmatrix} \begin{bmatrix} \beta\text{place} \\ -\text{sonorant} \end{bmatrix}$$

Two rules were constructed, each motivated by one of the constraints.

(3) Devoicing

$$\begin{bmatrix} +\text{voice} \\ -\text{sonorant} \end{bmatrix} \rightarrow \begin{bmatrix} -\text{voice} \\ -\text{sonorant} \end{bmatrix} / \_ \begin{bmatrix} -\text{voice} \\ -\text{sonorant} \end{bmatrix}$$

(4) Place Assimilation

$$\begin{bmatrix} \alpha\text{place} \\ -\text{continuant} \\ +\text{sonorant} \end{bmatrix} \rightarrow \begin{bmatrix} \beta\text{place} \\ -\text{continuant} \\ -\text{sonorant} \end{bmatrix} / \_ \begin{bmatrix} \beta\text{place} \\ -\text{sonorant} \end{bmatrix}$$

Voicing assimilation and place assimilation are differently supported by the English lexicon and the productive processes of English. While regressive devoicing is attested in the world's languages, and adult English speakers have been shown to perceptually compensate for its use to a small degree (Darcy et al., 2009), it is not an active process

in English. In contrast, regressive nasal place assimilation is an optional process in compounds and phrases of English, and is obligatory within words and Level 1 derivations (Kiparsky, 1985), as reflected in the orthography of *impossible*. However, this process is less likely to happen when the trigger is a fricative; compare *impossible* to *infernal*. Nor does English assimilate velar nasals to labial place (Jun, 2004). As a result, we can expect variability in participants' learning of both processes — that is, neither rule is expected to be unlearnable or applied without exception. The rules are not equally learnable for English speakers, but the design of the experiment does not require them to be, as the experiment is designed to test the interaction between the rule participants are trained on and the constraint they are tested on, rather than a main effect of the rule.

These constraints and rules guided the construction of words in an artificial language. The language has a plural suffix -[fa], and singular nouns have no suffix. When pluralization is applied to stems ending in voiced obstruents, which in this language include only [b] and [d], *DF is violated and Devoicing applies. When pluralization is applied to stems ending in non-labial nasals, which in this language include [n] and [ŋ], *NF is violated and Place Assimilation applies.

The experiment has a between-participants design, where the participants are divided into two groups and each exposed to a different exposure and training phase. The test phase is the same across these two treatments.

Neither treatment sees any violations of either constraint. However, each treatment only sees direct evidence for one rule. Thus, for each treatment there is an *active rule* and a *hidden rule*. For a given treatment, participants are shown both the singular and plural form of stems that undergo the active rule, but only a singular or a plural for each stem that would undergo the hidden rule. Thus, the application of the hidden rule is neither confirmed nor denied. There is phonotactic evidence for the constraint that motivates the hidden rule, because of the lack of violations of it

throughout the language, but there is no alternation-based evidence for the hidden rule.

The test phase then poses two-alternative forced choice questions concerning both constraints. For each constraint, there are questions pitting an apparently stem-internal violation of the constraint against a word that satisfies the constraint. Specifically, the constraint-satisfying word is identical to the constraint-violating word except for the first segment of the constraint violation — it is as if the rule has applied to repair the violation.

The dependent variable measured in this experiment is the probability of choosing a constraint-violating word in the test phase. If alternation-based evidence can affect one's phonotactic grammar, there should be an interaction between the treatment a participant is given and the constraint being tested, so that when participants are trained to apply a certain rule, they disprefer violations of the constraint that motivates that rule more than participants who were not trained to apply that rule.

3.2.1.1 Participants

One hundred participants were recruited from Mechanical Turk and paid for their participation. They were all located in the United States and claimed to be over 18 years old and native speakers of English. As in Experiment 1, participants were only run during the hours of noon and 5pm Eastern time on weekdays. Participants were paid $2.25, as the experiment was predicted to take up to 15 minutes.

Participants were excluded from the analysis based on native speaker status, whether they seemed to be paying attention, and whether they seemed to have learned the rules in the training session.

Native speaker status was assessed as in Experiment 1, based on questions about their native language and the language they speak at home. No participants were

excluded on this basis, as all responded that they are native and regular speakers of English.

Participants were considered inattentive if they answered too quickly or chose the option on one side of the screen too consistently. Reaction times under 50ms are likely due to bots, so any participants with such short times were excluded. Participants who chose the option on the left or the option on the right more than 90% of the time were also excluded. These criteria were applied and one participant was excluded based on answer speed.

In place of catch trials, I further assessed attention and success at the task through participants' performance in the training phase. The training phase repeated a maximum of five times. If they did not correctly answer 80% of the graded questions in a training round before getting to the fifth round, they were assumed to have not learned to apply the rule their language supports, and were excluded from the main analysis. Thirty-six participants were excluded on this basis.

In total, 63 participants were included in the analysis.

3.2.1.2   Inventory

The words of the artificial language used in this experiment were presented orthographically. The inventory of the language consisted of the following letters:

(5)   Inventory

    a.   Trigger for both constraints: *f*

    b.   Triggers for *DF: *b*, *d*

    c.   Repairs for *DF: *p*, *t*

    d.   Triggers for *NF: *n*, *ng*[1]

    e.   Repair for *NF: *m*

---

[1]Participants were instructed that in this language, *ng* is always pronounced as in *singer*, never as in *finger*.

      f.   Others: *l, s, a, e, i, o, u*

3.2.1.3   Exposure Phase

The first phase of the experiment after instructions were given was the exposure phase. The purpose of this phase was to begin teaching the participants the phonotactic patterns and the active rule without yet testing their memory. The task in this phase was to simply type the word or words of the artificial language that were displayed on the screen.

This phase consisted of three kinds of items: singular only, plural only, and singular-plural. The singular-only items showed a singular noun from the artificial language. There were ten of these items, and their words all ended in triggers for the hidden rule.

The plural-only items, of which there were also ten, showed a plural noun that ended in a repair for the hidden rule. However, the stems used in singular-only items were never used in plural-only items. Thus, the evidence was consistent with the use of the hidden rule, but did not prove its application.

The singular-plural items, on the other hand, showed a singular noun and the plural version of that same noun. There were fifteen of these: ten showing the active rule, and five showing non-alternating stems. The items showing the active rule had a stem ending in a segment that triggers the active rule, and its plural form, showing that the active rule had applied. For instance, if the active rule was Devoicing, the singular would end in *b* or *d*, and the stem of the plural would end in *p* or *t*, respectively.

The items showing the non-alternating words had stems ending in *p, t, m, l,* or *s.* Their plurals violated no constraints and thus consisted of the faithful stem and the suffix -*fa.*

(6)   Exposure stimuli examples when Devoicing is the active rule

    a.   Singular-only (10): *lobon*

    b.   Plural-only (10): *funemfa*

    c.   Singular-plural, faithful (5): *teldus - teldusfa*

    d.   Singular-plural, alternating (10): *nemab - nemapfa*

(7)   Exposure stimuli examples when Place Assimilation is the active rule

    a.   Singular-only (10): *nemab*

    b.   Plural-only (10): *funepfa*

    c.   Singular-plural, faithful (5): *teldus - teldusfa*

    d.   Singular-plural, alternating (10): *lobon - lobomfa*

The exposure stimuli were generated using a CV(C)CVC template. The consonants of the inventory were evenly distributed over the stimuli in the first consonantal slot and also in the second mandatory consonantal slot, with the exception that word-initial *ng* was swapped with the consonant that had been placed in that word's second onset, to avoid distracting participants with an ungrammatical word-initial *ng*. The optional medial consonant was placed in two of the singular-plural faithful stimuli and three of the singular-plural alternating stimuli (for each kind of alternation), in order to show participants that word-internal consonant clusters are allowed in this language, since they appear in the test words. They always consist of consonants that are not triggers for a specific rule, so that their presence doesn't influence answers to the test questions. The final consonant is dictated by the type of stimulus, as it is the one that may undergo rule application. Both vowel slots were filled by evenly distributing the vowels of the inventory over words and positions. The stimuli used in other portions of the experiment were generated similarly: positions whose identities were not dictated by the needs of the experimental design were filled via uniform distribution of sounds from the inventory, except that *ng* was kept out of word-initial position.

71

The instructions at the beginning of the exposure block were as follows:

> In this part of the experiment, you will learn words from a made-up language.
>
> Sometimes you'll see one word, and sometimes you'll see two: the singular version of the word first, and then the plural version.
>
> To help yourself catch on to this new language, type the words you see into the text box. We recommend pronouncing them out loud, too. In this language, "ng" is always pronounced as in "singer", never as in "finger".

Each exposure question had the following text: "Please write this word in the text box and pronounce it to yourself:" followed by one word or an appropriately pluralized version of that text followed by a singular-plural pair of words in the format "lobon - lobomfa". Below was a text box. No pictures or meanings were given to indicate the number of the nonce words, but as participants saw singular-plural pairs in their predictable order, they may have been able to recognize the plural suffix elsewhere.

All stimuli are given in Appendix D. The materials were generated and then read into a Python script that used Speriment to generate a website that was then launched on Mechanical Turk using PsiTurk. The Python script is in Appendix C.

3.2.1.4   Training Phase

After the exposure phase, participants went through a training phase. The goal of this phase was to ensure that participants had learned the active rule. The task was to choose the correct of two plural forms for a given singular form.

The phase consisted of thirty items: ten for the active rule, ten for the hidden rule, and ten for non-alternating fillers.

The instructions for this phase were as follows:

> Now we'll focus on learning singulars and plurals a bit more. You'll see the singular version of a word and two possible plurals. Choose the one you think is correct for this language. Sometimes you'll then be given the correct answer afterwards, to help you learn. Other times you won't see how you did.

The length of this phase depends on your accuracy. We've found that people who take their time on these questions actually finish this phase **much faster** than those who rush through the questions. Take your time and pronounce the correct answers out loud when they're given to help you learn how this language sounds.

The active rule items showed a singular noun ending in a trigger for the active rule, and presented two potential plural forms: one applying the active rule and one failing to apply it. The participant's response was considered correct if he or she chose the form which applied the rule. Correct responses were followed by a page saying "Correct!" and showing the correct singular-plural pair. Incorrect responses were followed by a page saying "No, the correct pairing is" followed by the correct singular-plural pair.

The filler items showed a singular noun ending in l or s and presented a plural form with a faithful stem and a plural form where the final consonant of the stem had been changed from l to s or vice versa. The faithful choice was considered correct. The responses were followed by the same kind of feedback as described for the active rule items.

The hidden rule items showed a singular noun ending in a trigger for the hidden rule and two options for its plural, one with a faithful stem and one having undergone the hidden rule. The participants are not taught whether the hidden rule applies or not, so no feedback was given for these items and they were not considered correct or incorrect.

The participant's score on the active rule and filler items was calculated by Speriment. If the participant had answered less than 80% of the graded questions correctly, the training block would repeat, with the question order newly shuffled. This would continue until either the participant passed the 80% mark or the block ran five times, at which point the participant would continue on to the testing phase.

3.2.1.5 Testing Phase

The testing phase was intended to test phonotactic judgments of both constraints. It was identical for participants in both treatments. The task was to choose which of two words sounded more like it belonged in the artificial language. The two words were minimal pairs. Neither contained a plural suffix; rather, both had an *f* in the middle of the word. The words differed in the segment preceding the *f*. In fillers, they differed randomly. In test items, one of the options would have a segment that triggered a given constraint and the other option would have a segment that repaired that constraint.

(8)   Example options on a test item for *DF (20)

    a.   madfas

    b.   matfas

(9)   Example options on a test item for *NF (20)

    a.   mangfas

    b.   mamfas

(10)   Example options on a filler item (10)

    a.   sulfen

    b.   susfen

As you can see in the examples above, the same word frames were used to create test items for both constraints, while different ones were used for fillers. For each constraint, there are two triggering segments. Each triggering segment was used in half of the word frames for a given rule. Within the half that used one triggering segment for a given rule, half used one triggering segment for the other rule and half used the second triggering segment for the other rule.

3.2.2 Results

The hypothesis that alternations affects phonotactics predicts an interaction between training condition and testing condition. Specifically, if learning a rule makes people disprefer the kind of violation that is removed by that rule, then we predict that participants will prefer violations of the constraint motivating the hidden rule more than they prefer violations of the constraint motivating the active rule, regardless of the rule that was active. The interaction plot in Figure 3.1 shows strong evidence of such an interaction. Recall that *DF motivates Devoicing and *NF motivates Place Assimilation. When training and testing "match," that is, a participant is trained on a rule and tested on the constraint that motivates that rule, they disprefer violations more than when training and testing do not match. The same means are given, along with standard deviations, in Table 3.1.

Table 3.1. Percent of times a constraint violation was chosen by condition in Experiment 3.

Rule Trained On	Constraint Tested On	Mean	Standard Deviation
Devoicing	*DF	51	50
Devoicing	*NF	58.2	49.4
Place Assimilation	*DF	63.3	48.2
Place Assimilation	*NF	40.9	49.2

The two high points in the plot represent the preferences of participants for violations that motivate hidden rules. These can be viewed more or less as participants' baseline preference for such violations due to their background knowledge as an English speaker. The baseline preference for violations of *DF is higher than the baseline preference for violations of *NF, which is expected given the English place assimilation rule. The absolute values of the slopes of the two lines are also different. This can be understood to mean that those who were encouraged to apply Place Assimilation, a rule similar to one they already know, responded more strongly than those who were encouraged to apply a novel rule, Devoicing.

For the present study, these differences across groups are relevant only insofar as their interpretation reassures us that the participants behaved in a reasonable manner, indicating that the task worked as expected. The point of interest in this plot is that the slopes of the lines are opposite, showing that the type of violation participants preferred more depended on the type of training they received, and thus supporting the hypothesis that training, that is, knowledge of alternations, can affect phonotactic judgements.

Interaction of Rule and Constraint
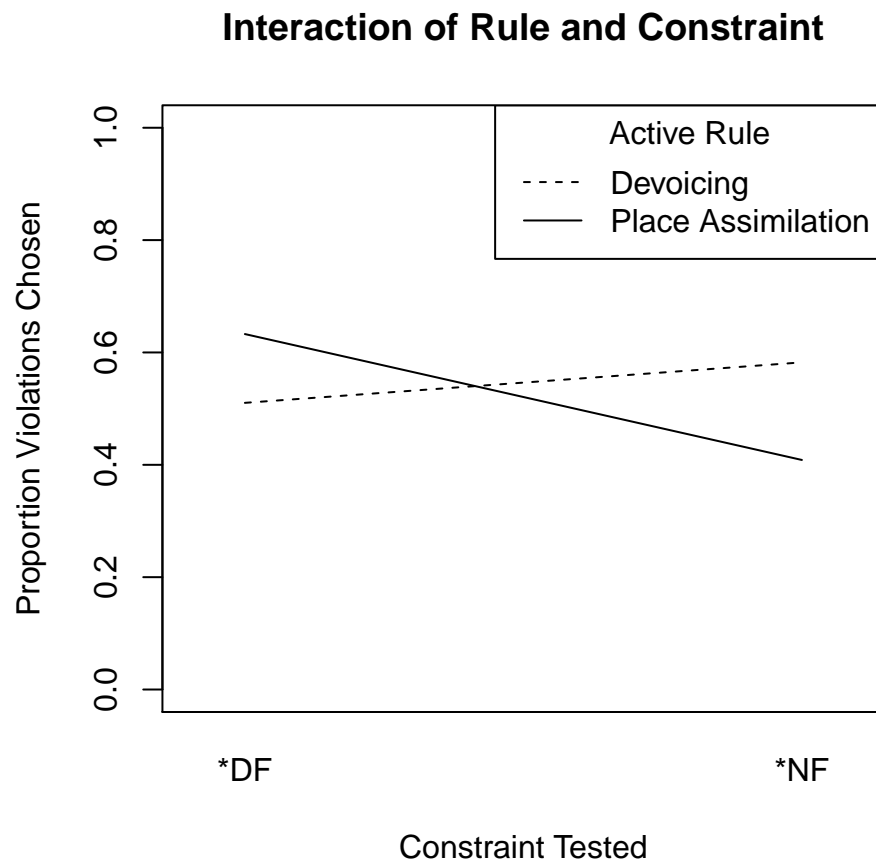
Figure 3.1. Interaction between effect of rule training and constraint testing in Experiment 3.

For a more detailed view of the results, Figures 3.2 and 3.3 show violin plots of data. The former shows a violin plot of the percent of times each participant chose a constraint violation in each condition. Figure 3.3 shows the same violin plot aggregated by item rather than by participant. As in the interaction plot, the violin plots show that a match between training and testing generally produces lower preferences for violations than a mismatch does. In these figures, Devoicing is abbreviated 'D' and Place Assimilation is abbreviated 'PA.'

Preference for Violation by Participant
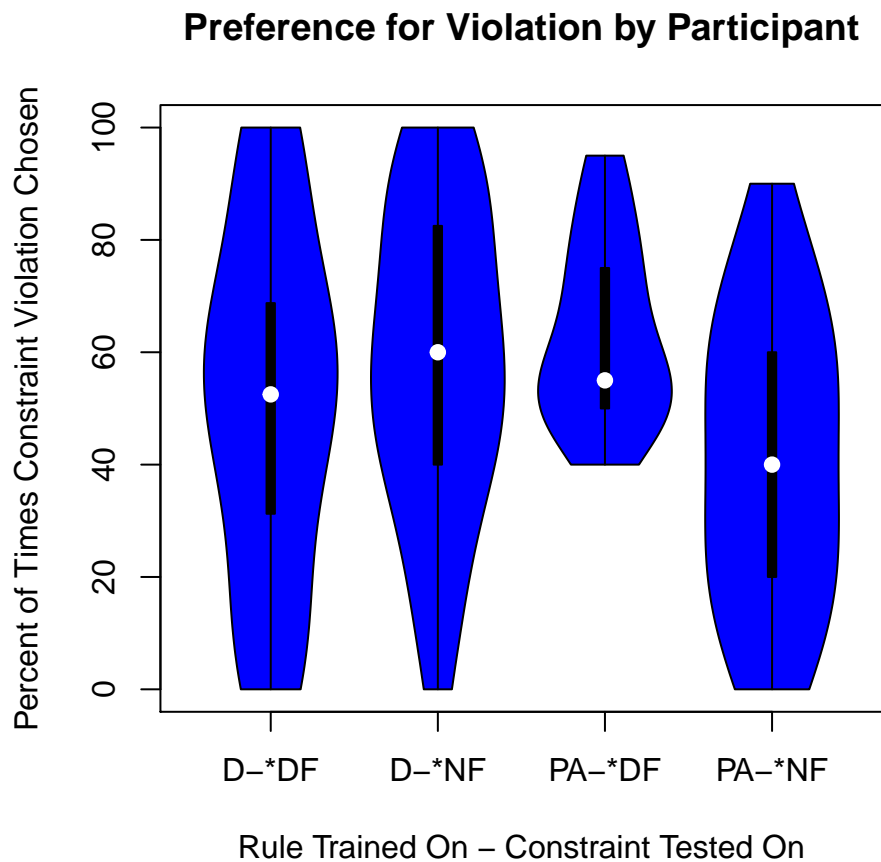
Figure 3.2. Distribution of by-participant violation preference for each condition in Experiment 3.

Figure 3.2 shows that the effect of training on Devoicing is subtle; the first two violins are fairly similar looking, although the median and interquartile range show a higher preference for violating *NF than *DF, as predicted. The effect of Place Assimilation training is more obvious. The distribution of participant means for those trained on Place Assimilation when they were tested on *DF is differently shaped than the other three distributions. This shows that no participants chose *DF-violating forms at a low rate after being trained on Place Assimilation. It is useful to bear in mind that the smooth distributions drawn on violin plots represent counts of discrete points; accordingly, it's normal for these plots to end somewhat abruptly rather than tapering to a fine point. Only six participants are represented in the portion of the Devoicing-*NF violin below 40%. However, the shape of the Place Assimilation-*DF violin is striking, and may be due to the fact that this condition is the one in which neither training nor English language knowledge militate against violations.

Figure 3.3 once again reflects the predicted interaction. Of note are the distributions for those who were trained on Place Assimilation. When tested on *DF, a very few items elicited a preference for unfaithful forms across participants. So even though no participants consistently answered this way across items, the variability across items was fairly high. When these participants were tested on *NF, so that both their native grammar and their training encouraged them to avoid violations, no items elicited high preference for violations. As in the grand means and the by-participant data, the effect of Devoicing training was smaller than the effect of Place Assimilation training.

A logistic mixed effects model was fitted to the data using the lme4 package (Bates and Maechler, 2009) in R (R Development Core Team, 2011). It included random slopes and intercepts for participants and items. The fixed effects were the training condition, the testing condition, their interaction, and the side of the page the constraint-violating word was presented on. The complete formula is given in (11).

Preference for Violation by Item
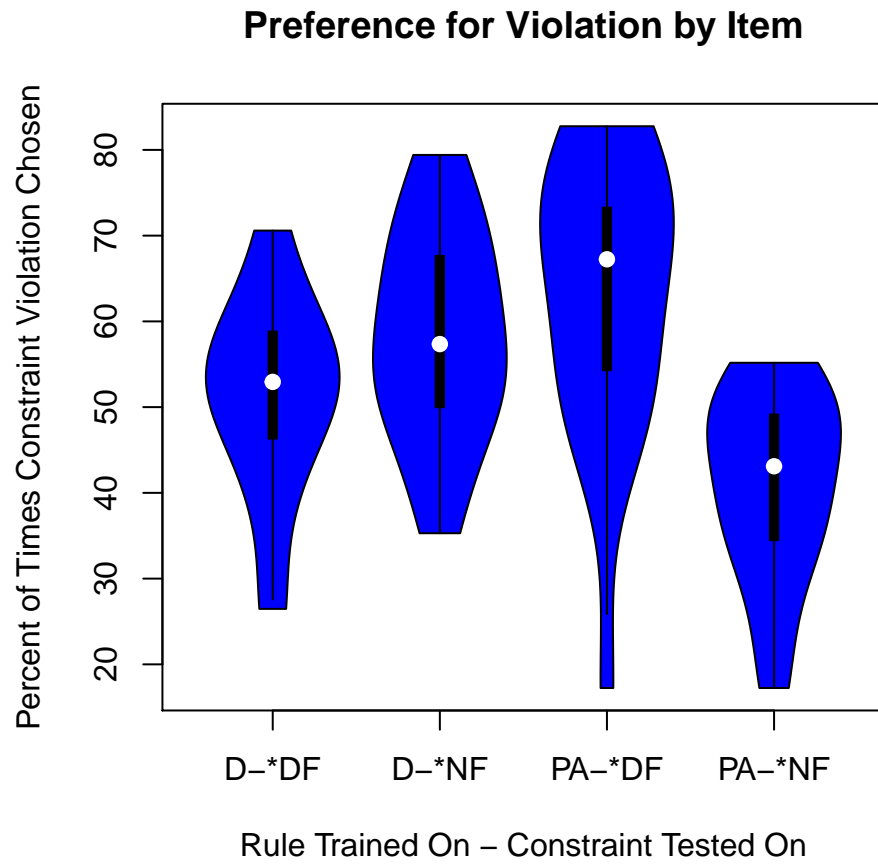
Figure 3.3. Distribution of by-item violation preference for each item in Experiment 3.

Table 3.2 shows the coefficients found for this model. It shows that the main effects of training condition and testing condition were not significant. However, as predicted, their interaction was significant, at the $\alpha = 0.001$ level. Another significant effect was that of the positioning of the options. Participants were more likely to choose the constraint-violating option if it appeared on the left.

(11)   Mixed effects model formula

ChoseViolation $\sim$ Permutation $*$ Violates + ViolationPosition + (1 | Participant) + (0 + Permutation | Participant) + (0 + Violates | Participant) + (0 + Permutation:Violates | Participant) + (1 | Item) + (0 + Permutation | Item) + (0 + Violates | Item) + (0 + Permutation:Violates | Item)

Table 3.2. Coefficients of mixed effects model for Experiment 3.

Factor	Estimate	p-value
Intercept	0.17	0.2906537
Active Rule	-0.1	0.4911305
Constraint Tested	-0.2	0.2004161
Active Rule:Constraint Tested	-0.42	0.0018844
Violation Position	-0.42	2.055024×10^{-5}

3.2.3   Discussion

The results support the hypothesis that knowledge gleaned from alternations can affect the phonotactic grammar, and motivate research on phonotactics that takes the presence of alternations into account. The modeling of phonotactics alone may still be useful as a methodological abstraction, but this study suggests that it will be important to consider how this simplification may skew the results. This study also bears on work on derived environment effects. Such effects, reviewed in Wolf (2008), occur when a rule applies at a morpheme boundary but not in a monomorphemic context. By showing the generalization of a rule learned at a morpheme boundary to a presumably monomorphemic context, these results suggest that the presence

of derived environment effects may not be the baseline hypothesis of the language learner.

The primary weakness of this experiment is the reliance on feedback to train participants on the rule. The problem with feedback is that it is necessarily given asymmetrically. The design of the experiment hinges on one rule being active while the other is hidden, so feedback cannot be given for the hidden rule. Yet, feedback may increase participants' familiarity with the forms that are shown to be correct or incorrect, which could have an effect on the phonotactic grammar directly. This effect is subtle, because the test questions do not pit one constraint against the other, and so a feedback bias is partially avoided. For concreteness, here are the types of forms shown in the feedback given in the Devoicing treatment:

(12)   Types of feedback in Devoicing treatment

     a.   Irrelevant fillers: *beful* ∼ *befulfa*

     b.   Faithful voiceless fillers: *pidep* ∼ *pidepfa*

     c.   Faithful labial fillers: *dulim* ∼ *dulimfa*

     d.   Devoicing training: *sapod* ∼ *sapotfa*

The fillers are the same across treatments, so it is the Devoicing training words that introduce an asymmetry, as they are replaced by Place Assimilation training words in the Place Assimilation treatment. The test questions pit constraint-satisfying words against constraint-violating words for a particular constraint, such as *ludfum* vs. *lutfum* and *lunfum* vs. *lumfum*. A participant in the Devoicing condition will have seen feedback containing stem-final *d* and *t*, and no non-filler feedback containing stem-final *n* or *m*. However, if we consider bigrams, the participant has seen non-filler feedback containing *tf*, and no non-filler feedback containing *df*, *mf*, or *nf*.

Thus, if participants update their phonotactic grammars of the artificial language based not just on the presence of the test words but also on the presence of feedback

for the test words, there is a confound that should cause participants to prefer forms that satisfy the constraint that motivates their active rule. It is unclear if this effect is plausible, because token frequency, which is increased by feedback while type frequency is held constant, has not been found to be predictive of phonotactic judgments (Albright, 2008b). However, feedback may simply amplify attention.

The best way to assess this possibility with the data from this experiment is to look at the responses from participants who only spent one iteration in the training phase; they had less exposure to feedback than any other participants, and may well have exited the exposure phase already having learned the rule.

The plot in Figure 3.4 shows the interaction plot for only those participants who met the criterion in the first iteration of training, and Figure 3.5 shows the interaction plot for the participants who met the criterion after two, three, or four iterations. The slopes of the lines in the interaction plot are not identical, but they have the same sign, showing the same direction of change across conditions. The difference appears to be primarily that longer training in Devoicing lowered the preference for violations of *DF; in other words, Place Assimilation appears to not only have a larger effect on test performance, but also to have that effect more quickly. The presence of a trend towards the predicted interaction in both figures suggests that feedback is not the sole reason for the presence of the effect in the experiment as a whole. However, feedback is given in the first iteration, so this possibility cannot be ruled out completely. Furthermore, a statistical test of the interactions over these smaller participant pools cannot be carried out because the models lack sufficient data to converge, given their complex random effects structure.

3.3 Experiment 4

In order to address the confound in Experiment 3, where the presence of feedback for words where training and testing conditions match but not for words where they

Interaction After One Iteration
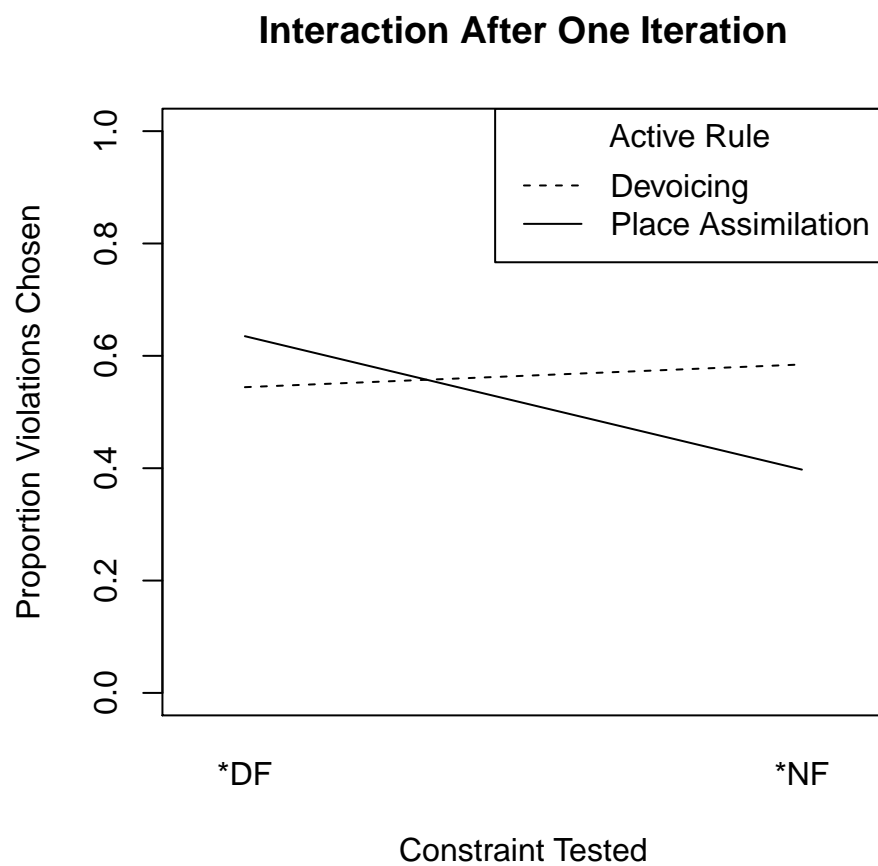
Figure 3.4. Interaction between rule trained on and constraint in question for participants who took only one training iteration.
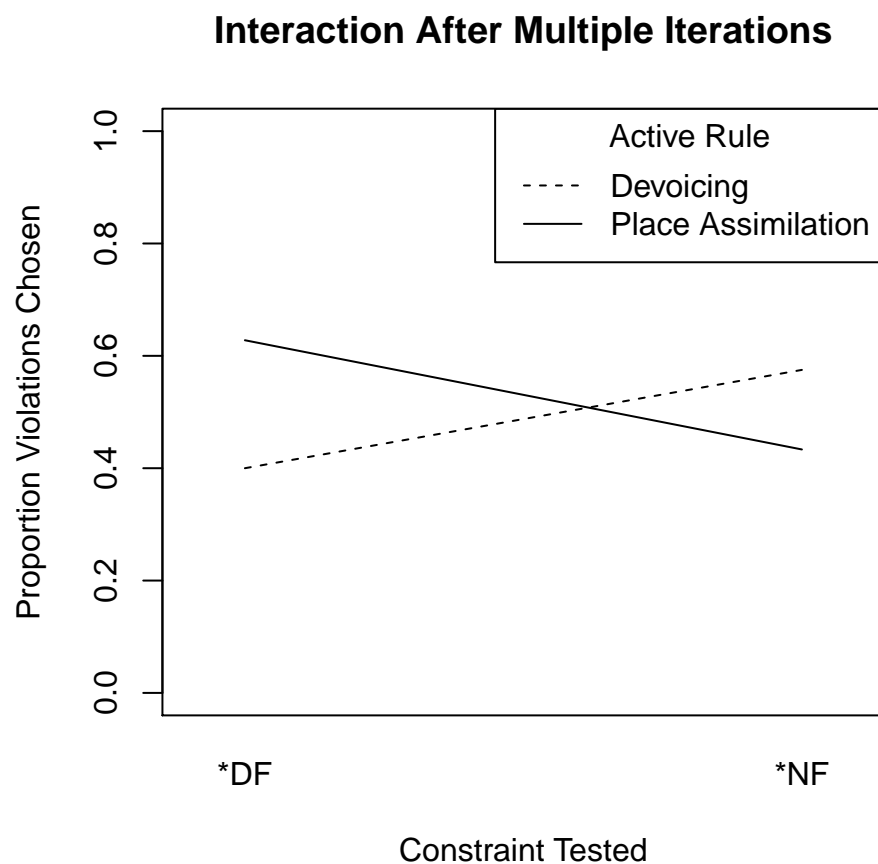
Figure 3.5. Interaction between rule trained on and constraint in question for participants who took more than one training iteration.

mismatch, a second experiment was performed. Experiment 4 avoids this confound by removing feedback from the design, resulting in an experiment with an exposure phase, one iteration of training where no feedback is given, and a testing phase.

3.3.1 Method

Experiment 4 is very similar to Experiment 3; the parts of the method that were different are described below.

3.3.1.1 Training Phase

Recall that the training phase in Experiment 3 consisted of questions where a singular form was presented and the participant was asked to choose between two possible plural forms, and then, for some trials, given feedback naming the correct singular-plural pair. The training phase in Experiment 4 used the same stimuli and question format, but eliminated all feedback. Because the lack of feedback made it unlikely that repeated iterations would improve performance, only one iteration was given, regardless of accuracy rate. Due to the lack of feedback, this phase was not truly a training phase, and is called that only for comparison with Experiment 3. Rather, this phase was used to determine which participants had learned the rule sufficiently that they should be included in the analysis. It is possible, though, that participants organized and solidified their knowledge of the rule by answering the questions in this phase.

3.3.1.2 Testing Phase

The testing phase was identical except that, in order to increase the validity of the assumption that participants were treating test words as underived, the instructions to the test phase specified that unlike the words seen previously, these did not have any suffixes added.

3.3.1.3 Participants

Participants were once again run on Mechanical Turk. Because the experiment was shorter, they were paid $1.10 each, under the assumption that the experiment would take about 7 minutes to complete. All exclusion criteria from Experiment 3 were used except the number of iterations, which was not applicable because multiple iterations were not possible in Experiment 4. These criteria resulted in two participants whose native languages were not English being excluded from the analysis.

In place of number of iterations as a metric of whether participants learned the alternation, percent correct answers in the single training iteration was used. In Experiment 3, 46 of 99 participants reached criterion in their first iteration of training. In order to get a similar amount of usable data, 200 participants were run on Experiment 4. However, due to the lack of feedback, which could help participants learn even within one iteration, a lower percentage of participants reached the original criterion of 80% correct, yielding 57 participants. To increase the likelihood of having sufficient data, the criterion was lowered to 70% accuracy. This decision was made before analyzing any data to avoid anti-conservative effects due to data peeking. This resulted in 80 participants being included in the analysis.

3.3.2 Results

Table 3.3. Percent of times a constraint violation was chosen by condition in Experiment 4.

Rule Trained On	Constraint Tested On	Mean	Standard Deviation
Devoicing	*DF	52.1	50
Devoicing	*NF	58.6	49.3
Place Assimilation	*DF	55.7	49.7
Place Assimilation	*NF	49.4	50

Figure 3.6 shows the interaction plot for Experiment 4. Once again, we see that the slopes of the lines have different signs, indicating that training on a rule decreased

preferences for violating the constraint that motivates that rule. Compared to Experiment 3, participants trained on Place Assimilation behave less differently on the two constraints.

Interaction of Rule and Constraint
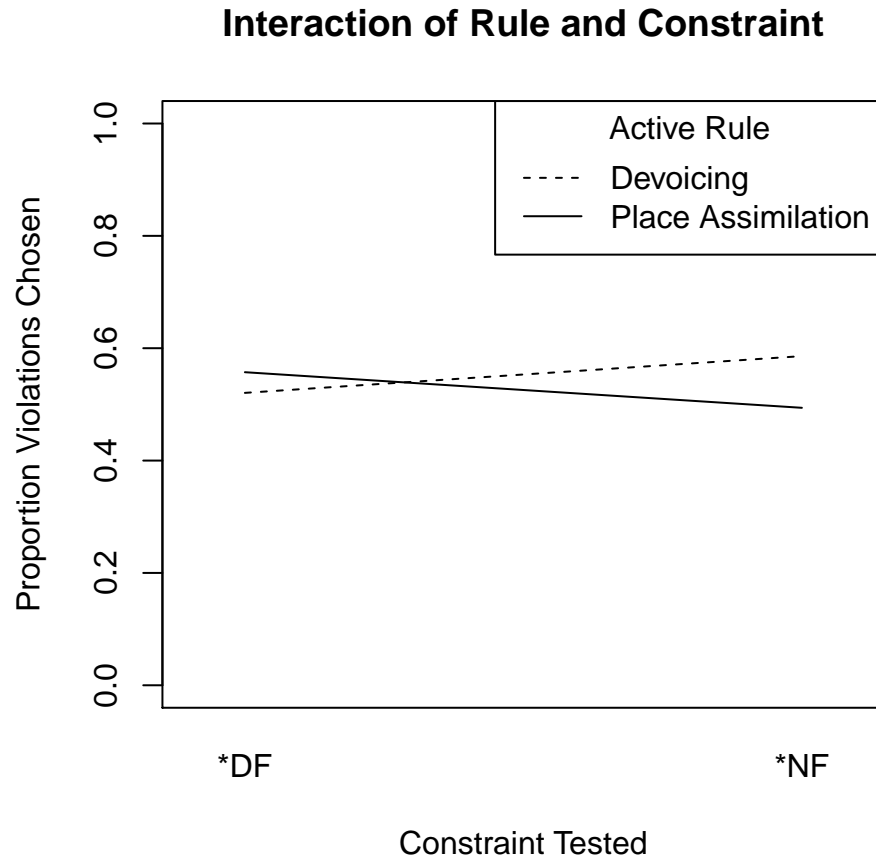
Figure 3.6. Interaction between rule training and constraint testing in Experiment 4.

Figure 3.7 shows a violin plot of the percent of times each participant chose a constraint violation in each condition. It shows a clear trend towards an effect of training in the Devoicing condition; the preference for violations is lower when the constraint tested is *DF, motivated by the active rule, than when it is *NF. This trend is less clear in the Place Assimilation condition.
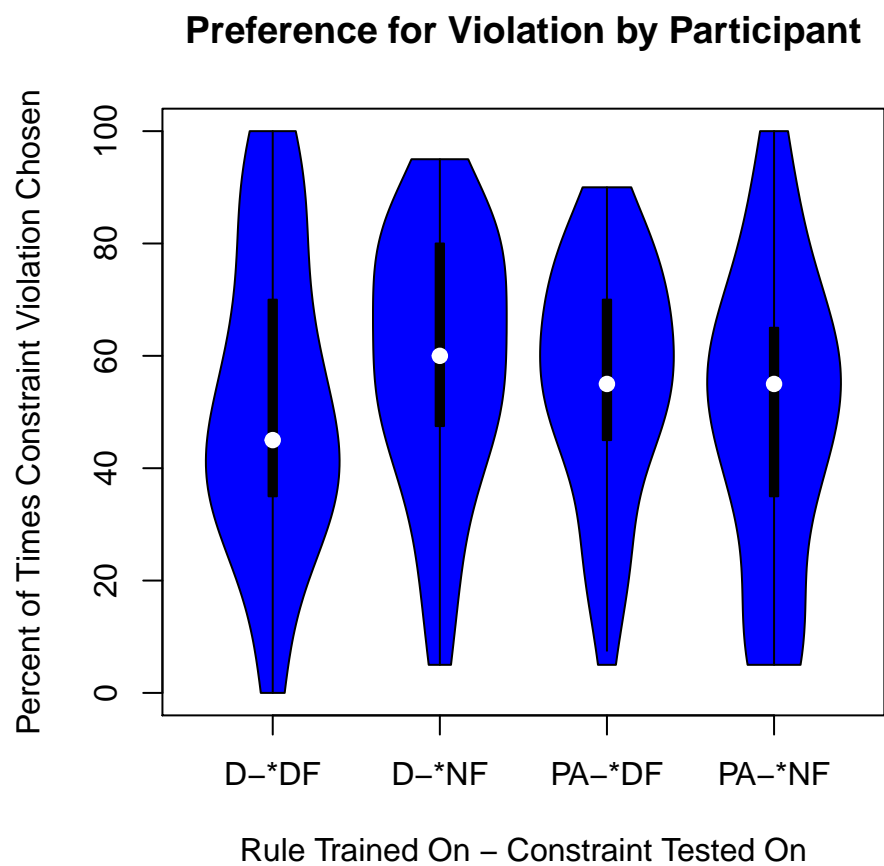
Figure 3.7. Distribution of by-participant violation preference for each condition in Experiment 4.

Figure 3.8 shows the same violin plot aggregated by item rather than by participant. In this plot, the trend is clearer in the Place Assimilation condition. As in Experiment 3, items testing *DF among participants trained in Devoicing have a long lower tail, showing that a few items are preferred without the constraint violation. In the Devoicing condition, the trend is more subtle, but still leans in the predicted direction.

Preference for Violation by Item
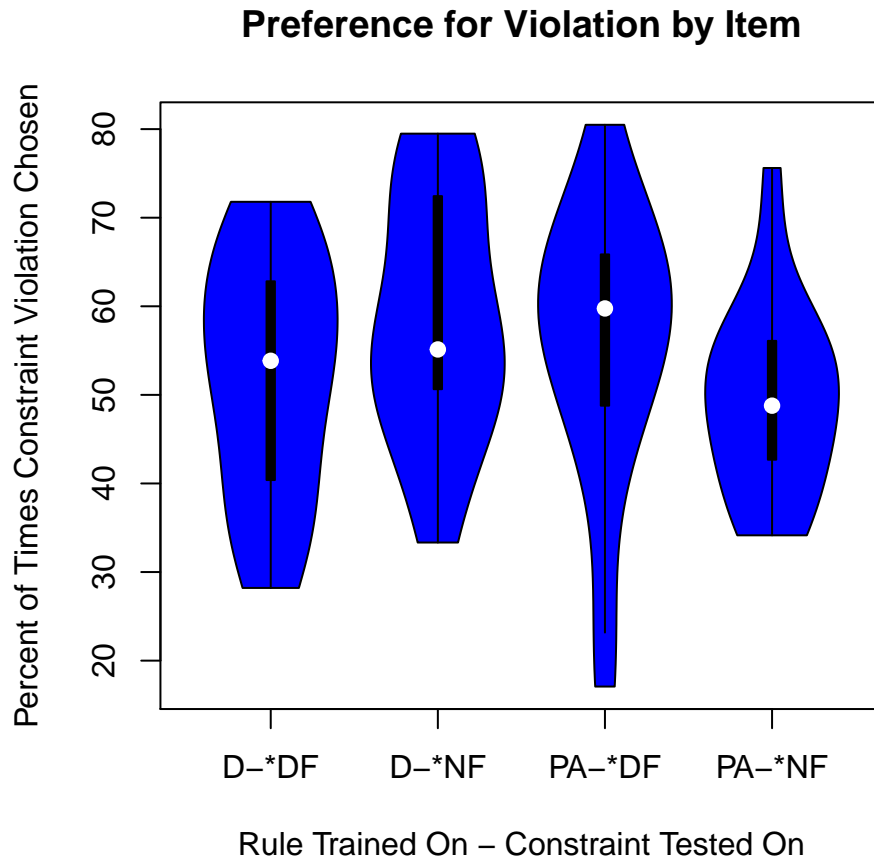
Figure 3.8. Distribution of by-item violation preference for each item in Experiment 4.

The same model structure that was used to analyze the data from Experiment 3 was used to analyze this data, although some random effects were omitted to allow

the model to converge. The resulting formula is given in (13). The coefficients found by the model are shown in Table 3.4.

(13)   Mixed effects model formula

ChoseViolation $\sim$ Permutation $*$ Violates $+$ ViolationPosition $+$ (1 | Participant) $+$ (0 $+$ Permutation | Participant) $+$ (0 $+$ Violates | Participant) $+$ (1 | Item) $+$ (0 $+$ Permutation | Item)

Table 3.4. Coefficients of mixed effects model for Experiment 4.

Factor	Estimate	p-value
Intercept	0.2	0.1574127
Active Rule	-0.08	0.4456613
Constraint Tested	-0.01	0.9693086
Active Rule:Constraint Tested	-0.16	0.1040932
Violation Position	-0.34	4.1330069×10^{-5}

As before, there is a significant effect of the position of the constraint-violating option on the page, and the main effects of training and testing are not significant. Unlike in Experiment 3, the interaction of training and testing does not reach significance. The trend is in the predicted direction, but has not been confirmed statistically.

3.3.3   Discussion

The results of Experiment 4 are inconclusive. The data show a trend in the same direction as Experiment 3, but the predicted interaction between training and testing is not significant. This could mean that the finding of Experiment 3 was indeed due to feedback, but it could also mean that without feedback, participants learned the generalization less well, resulting in noisier results. When variance increases, all else held equal, power decreases, so it is plausible that Experiment 4 failed to find a significant result due to a lack of power.

90

3.4 General Discussion

These experiments offer mixed support for the hypothesis that information about alternations is used in making phonotactic judgments. Experiment 3 found support for the hypothesis, but may have inadvertently drawn participants' attention to words that biased them towards giving such results. Experiment 4 eliminated feedback, and as a result the active rule was learned at a far lower rate. The criterion used to determine whether a participant had learned the rule sufficiently to be included in the analysis was lowered in an attempt to increase power, and this may have increased the noise in the results. The predicted interaction between training and testing was not significant, in contrast to Experiment 3. Further experimentation is needed to determine whether Experiment 4 merely lacked power or failed to show an effect because it the effect had been due to a confound.

Nevertheless, the trend found in Experiment 4 makes it worthwhile to consider the possibility that alternation-based knowledge affects phonotactics and spell out the implications if clearer evidence is found with further study. It would not show that alternations and phonotactics are necessarily computed from the same, monolithic system, as Optimality Theory asserts, although that may be the case. Rather, it would suggest that whether these types of knowledge are encoded in one system or in two, there is at least a pathway for communication between them. Specifically, it would argue for the flow of information from alternations to phonotactics. Although the findings of Pater and Tessier (2003) are suggestive, this study cannot confirm the flow of information from phonotactics to alternations. However, of the two directions of information flow, that from alternations to phonotactics is the more surprising. Since phonotactic knowledge is likely to be acquired first, it may be helpful in acquiring alternations. The need for alternations to affect phonotactics is less obvious; phonologists may find it more plausible that the two forms of knowledge are in two-

way communication than that alternations affects phonotactics but the reverse is not true.

CHAPTER 4

CONCLUSION

This dissertation has aimed to push forward the study of phonotactics with experimental results as well as aid in future advances with software to facilitate further experimentation. Chapter 1 presented Speriment, experiment creation software that is particularly tailored to phonotactic research in that it includes the specific features needed to run judgment and artificial language learning experiments. By using this package, experimenters can reduce the amount of time they spend creating and running experiments, and easily share their designs and results with others.

Chapter 2 presented experiments that found and replicated an effect showing that constraint violations do not accumulate linearly, but rather, a violation in the presence of another violation has a smaller effect on word acceptability than it does in isolation. This finding suggests that phonotactic modeling should not be done in linear Harmonic Grammar, or other frameworks that predict linear accumulation of violations. It also reinforces the view that phonotactic modeling should not be done in Optimality Theory, where violations do not accumulate at all across constraints. It offers preliminary support for the use of Maximum Entropy Grammar in modeling phonotactics, and any other models that predict subadditive accumulation of violations.

Narrowing down the space of constraint-based models is useful in several ways. First, it gives researchers who would like to improve existing constraint-based models a clearer starting point. Building on Maximum Entropy Grammar is likely to be more fruitful than building on models that incorrectly model violation interaction.

Second, it gives researchers who would like to propose new models a desideratum for their model and a suggested point of comparison between their model and Maximum Entropy Grammar. The desideratum is that their model predict subadditive cumulativity of violations. The point of comparison is in the specific ways that a new model and Maximum Entropy Grammar create this subadditivity. The finding that the acceptability loss from two violations is less than the acceptability loss from each violation is isolation is consistent with many methods of combining violations. Maximum Entropy Grammar uses exponentiation to create an effect of diminishing marginal penalties, but other methods are possible. New models of constraint-based phonotactics could be compared against Maximum Entropy Grammar using more sensitive tests in order to learn more about the nature of violation cumulativity.

Finally, this finding gives researchers who would like to compare constraint-based models against other classes of models more justification in choosing Maximum Entropy Grammar (or another subadditive model) over Optimality Theory or linear Harmonic Grammar as a representative of constraint-based phonotactics. A theory comparison is only fair if each is represented as well as possible, and Experiments 1 and 2 provide researchers with one possible test for choosing a representative of constraint-based phonotactics.

Chapter 3 presented experiments that investigated whether learning a phonological alternations affects phonotactic judgments. Experiment 3 found support for this hypothesis. Experiment 4 attempted to replicate this effect with a different style of training, in order to eliminate a possible confound. It found a nonsignificant trend in the predicted direction, motivating further research to distinguish between the discovery that a confound was responsible for the effect of Experiment 3, and merely a lack of power in Experiment 4. Because both experiments were designed using Speriment, their designs are easy to adjust and replicate, making this next step accessible to other researchers. If the hypothesis supported by Experiment 3 holds upon fur-

ther investigation, the finding will be compatible with the assumption often made in constraint-based grammars that phonotactics and alternations are encoded in one constraint set. It would suggest that models that account for phonotactics and alternations separately should at least have a mechanism for communication between the two repositories of knowledge.

This dissertation shows how experiments can be efficiently created and used to shed light on questions about the basic architecture of the phonological grammar. The question left open by Experiment 4 is a common situation in experimental work, and highlights the importance of allowing other researchers to inspect, tweak, and replicate experiments before drawing conclusions. Speriment complements the findings of this dissertation by providing a means for others to build upon them and question them with accuracy and efficiency.

APPENDIX A

EXPERIMENTS 1 AND 2 CODE

A.1 Experiment 1 Code

```
from speriment import *


##### Wording #####


introduction = '''In this experiment, you will be shown made-up words.
Because they're not real words, they're spelled phonetically, so
sometimes the spelling will look weird even though the sounds are
fine. Try to focus only on how they sound. Pronounce the words to
yourself.
<br><br>
You'll be asked to say whether the word could be a
word of English. You don't have to say 'yes' and 'no' an even number
of times. Just answer whatever you think based on the way the word
sounds.'''


instructions = '''During experimental questions, you will not be able
to use your mouse. Use the F key to choose the option on the left, the
J key to choose the option on the right, and the spacebar to move
forward. Make sure your CAPS lock is off.  At the end of the
experiment, you can use the mouse to answer some questions about
```

```
yourself.'''


question = '''Based on how it sounds, do you think this word could be
a word of English? <br><br>'''


native_language = "What is your native language?"


daily_language = "What language do you speak at home?"


age = "How old are you?"


sex = "What is your sex?"


strategy = '''What information or strategies did you use to choose
words in the experimental questions?'''


goodbye = '''Thank you for participating in this experiment! Make sure
to click the Complete HIT button on the next page.'''



##### Stimuli #####


rows = get_dicts('items_oct8.csv')


# condition, onset violation, coda violation
test_conditions = [('GG', 0, 0),
                   ('GB', 0, 1),
```

```python
                    ('BG', 1, 0),

                    ('BB', 1, 1)]


# good fillers

catch_words = ['lbifth',

               'krlisg',

               'psatrl',

               'tmozb',

               'zgdivl',

               'wlramr']


# bad fillers

catch_words2 = ['bressic',

                'claffer',

                'spellion',

                'frotion',

                'merly',

                'dellous']


##### Experiment structure #####


with make_experiment(IDGenerator()):


    intro_block = Block(pages = [

        Page(introduction,

             tags = {'PageType': 'Instructions'})])

    instruction_block = Block(pages = [
```

```
        Page(instructions, tags = {'PageType': 'Instructions'}),

        Page(instructions2, tags = {'PageType': 'Instruction'})])


test_page_groups = [[Page(
            [question, "<b>", row[test_condition], "</b>"],
            options = [
                Option("Yes", tags = {'Response': 1}),
                Option("No", tags = {'Response': 0})],
            tags = {'ItemID': str(i),
                    'OnsetViolation': onset,
                    'CodaViolation': coda,
                    'Vowel': row['vowel'],
                    'PageType': 'Test'},
            condition = test_condition,
            keyboard = True)
        for (test_condition, onset, coda) in test_conditions]
    for (i, row) in enumerate(rows)]


catch_trials = [Page([question, '<b>', word, '</b>'],
                    options = [Option('Yes',
                                        tags = {'Response': 1,
                                                'ExcludeMe': 1}),
                               Option('No',
                                        tags = {'Response': 0,
                                                'ExcludeMe': 0})],
                    tags = {'PageType': 'Catch'},
                    condition = 'worst',
```

```
                        keyboard = True)
                for word in catch_words]


catch_trials2 = [Page([question, '<b>', word, '</b>'],
                      options = [Option('Yes',
                                           tags = {'Response': 1,
                                                    'ExcludeMe': 0}),
                                 Option('No',
                                           tags = {'Response': 0,
                                                    'ExcludeMe': 1})],
                      tags = {'PageType': 'Catch'},
                      condition = 'best',
                      keyboard = True)
                for word in catch_words2]


# The test words come in four conditions, but the catch words
# don't, so I create groups of four identical catch words to avoid
# throwing off the Latin Square. I use the "new" method to give
# each copy a unique ID.
def make_catch_groups(catch_trial):
    catch_group = [catch_trial]
    for i in range(1, 4):
        new_trial = catch_trial.new()
        catch_group.append(new_trial)
    return catch_group


catch_groups = [make_catch_groups(catch_trial)
```

```
            for catch_trial in catch_trials + catch_trials2]


    # The test block includes the test pages and the catch pages.
    # They'll be shuffled together.
    test_block = Block(groups = test_page_groups + catch_groups,
                       latin_square = True)


demographics_block = Block(
    pages = [Page(age,
                  options = [Option(i) for i in range(120)],
                  ordered = True,
                  tags = {'PageType': 'Demographics'}),
             Page(sex,
                  options = [Option('male'), Option('female'),
                  Option('other')],
                  tags = {'PageType': 'Demographics'}),
             Page(native_language,
                  options = [Option()],
                  freetext = True,
                  tags = {'PageType': 'Demographics'}),
             Page(daily_language,
                  options = [Option()],
                  freetext = True,
                  tags = {'PageType': 'Demographics'}),
             Page(strategy,
                  options = [Option()],
                  freetext = True,
```

```
                    tags = {'PageType': 'Demographics'}),])


    thanks_block = Block(pages = [Page(
        goodbye,
        tags = {'PageType': 'Instructions'})])


    experiment = Experiment([intro_block,
                             instruction_block,
                             test_block,
                             demographics_block,
                             thanks_block])


    experiment.install('cumulativity_experiment')
```

A.2   Experiment 2 Code

```
from speriment import *


##### Wording #####


introduction = '''In this experiment, you will be shown made-up words.
Because they're not real words, they're spelled phonetically, so
sometimes the spelling will look weird even though the sounds are
fine. Try to focus only on how they sound. Pronounce the words to
yourself.
<br><br>
You'll be asked to say whether the word could be a word of English.
You don't have to say 'yes' and 'no' an even number of times. Just
```

answer whatever you think based on the way the word sounds.'''

instructions2 = '''Before we start, here are examples of the kind of
words you'll see.

blickity is the kind of word you might want to say "yes" to.
It's not an English word, but it sounds like it could be.

rzbesgathv is the kind of word you might want to say "no" to.
It's not an English word, and it doesn't sound like it could ever be
one.'''

question = '''Based on how it sounds, do you think this word could be
a word of English?

'''

native_language = "What is your native language?"

daily_language = "What language do you speak at home?"

age = "How old are you?"

sex = "What is your sex?"

strategy = '''What information or strategies did you use to choose
words in the experimental questions?'''

goodbye = '''Thank you for participating in this experiment! Make sure

```
to click the Complete HIT button on the next page.'''




##### Stimuli #####


rows = get_dicts('items_june17.csv')


# condition, onset violation, coda violation

test_conditions = [('GG', 0, 0),

                   ('GB', 0, 1),

                   ('BG', 1, 0),

                   ('BB', 1, 1)]


# bad fillers

catch_words = ['lbafthrizk',

               'kflisgwevr',

               'psafzotrl',

               'tmuhrizb',

               'zgdokpevf',

               'wlratlumr']


# good fillers

catch_words2 = ['bressic',

                'claffer',

                'spellion',

                'frotion',

                'merly',
```

```
                    'dellous']


##### Experiment structure #####


with make_experiment(IDGenerator()):


    intro_block = Block(

        pages = [Page(introduction,

                     tags = {'PageType': 'Instructions'})])


    instruction_block = Block(

        pages = [Page(instructions,

                     tags = {'PageType': 'Instructions'}),

                Page(instructions2,

                     tags = {'PageType': 'Instruction'})])


    # The test pages are in groups so I can use a Latin square.
    test_page_groups = [[Page([question,

                              "<b>",

                              row[test_condition],

                              "</b>"],

                            options = [Option("Yes",

                                             tags = {'Response': 1}),

                                       Option("No",

                                             tags = {'Response': 0})],

                            tags = {'ItemID': str(i),

                                    'OnsetViolation': onset,
```

```
                                  'CodaViolation': coda,

                                  'Vowel': row['vowel'],

                                  'PageType': 'Test'},

                        condition = test_condition,

                        keyboard = True)
         for (test_condition, onset, coda) in test_conditions]
     for (i, row) in enumerate(rows)]


# bad filler pages
catch_trials = [Page([question, '<b>', word, '</b>'],

                     options = [Option('Yes',

                                       tags = {'Response': 1,

                                               'ExcludeMe': 1}),

                                Option('No',

                                       tags = {'Response': 0,

                                               'ExcludeMe': 0})],

                     tags = {'PageType': 'Catch'},

                     condition = 'worst',

                     keyboard = True)
                for word in catch_words]


# good filler pages
catch_trials2 = [Page([question, '<b>', word, '</b>'],

                      options = [

                            Option('Yes',

                                   tags = {'Response': 1,

                                           'ExcludeMe': 0}),
```

```python
                                    Option('No',
                                          tags = {'Response': 0,
                                                  'ExcludeMe': 1})],
                            tags = {'PageType': 'Catch'},
                            condition = 'best',
                            keyboard = True)
                    for word in catch_words2]


# The test words come in four conditions, but the catch words
# don't, so I create groups of four identical catch words to avoid
# throwing off the Latin Square. I use the "new" method to give
# each copy a unique ID.
def make_catch_groups(catch_trial):
    catch_group = [catch_trial]
    for i in range(1, 4):
        new_trial = catch_trial.new()
        catch_group.append(new_trial)
    return catch_group


catch_groups = [make_catch_groups(catch_trial)
    for catch_trial in catch_trials + catch_trials2]


# The test block includes the test pages and the catch pages.
# They'll be shuffled together.
test_block = Block(groups = test_page_groups + catch_groups,
                   latin_square = True)
```

```
demographics_block = Block(

    pages = [Page(age,

                options = [Option(i) for i in range(120)],

                ordered = True,

                tags = {'PageType': 'Demographics'}),

            Page(sex,

                options = [Option('male'),

                        Option('female'),

                        Option('other')],

                tags = {'PageType': 'Demographics'}),

            Page(native_language,

                options = [Option()],

                freetext = True,

                tags = {'PageType': 'Demographics'}),

            Page(daily_language,

                options = [Option()],

                freetext = True,

                tags = {'PageType': 'Demographics'}),

            Page(strategy,

                options = [Option()],

                freetext = True,

                tags = {'PageType': 'Demographics'})])


thanks_block = Block(pages = [Page(

                            goodbye,

                            tags = {'PageType':

                                'Instructions'})])
```

```python
experiment = Experiment([intro_block,
                         instruction_block,
                         test_block,
                         demographics_block,
                         thanks_block])


experiment.install('cumulativity_experiment')
```

APPENDIX B

EXPERIMENTS 1 AND 2 STIMULI

B.1 Experiment 1

Table B.1. Test items in Experiment 1.

GG	BG	GB	BB
plag	plavb	tlag	tlavb
klep	klefp	dlep	dlefp
bloss	blofk	zboss	zbofk
glid	glishf	srid	srishf
pruss	prumth	zduss	zdumth
troom	trooshp	zgoom	zgooshp
skaff	skashk	shlaff	shlashk
brizz	brivg	shpizz	shpivg
drell	drefsh	shtell	shtefsh
grun	gruzv	shkun	shkuzv
spog	spozb	shnog	shnozb
stoob	stoosf	thloob	thloosf
krat	krathp	vlat	vlathp
sleb	slemg	vreb	vremg
shrin	shritp	znin	znitp
snock	snopk	zmock	zmopk
smeck	smenb	fmeck	fmenb
throoz	throozg	fnooz	fnoozg
flad	flathk	vnad	vnathk
skrep	skrebg	vmep	vmebg
swuff	swukp	thnuff	thnukp
splot	splogb	thmot	thmogb
frim	fridb	lrim	lridb
strool	stroothf	hrool	hroothf

B.2 Experiment 2

Table B.2. Filler items in Experiment 1.

High acceptability	Low acceptability
bressic	lbifth
claffer	krlisg
spellion	psatrl
frotion	tmozb
merly	zgdivl
dellous	wlramr

Table B.3. Test items in Experiment 2.

GG	BG	GB	BB
praz	pradb	zmaz	zmadb
skreff	skrefsh	shpeff	shpefsh
grot	gropk	shtot	shtopk
thrit	thrishk	lrit	lrishk
skun	skuthp	shnun	shnuthp
kloop	kloothk	thnoop	thnoothk
]snass	snathf	thmass	thmathf
krizz	krishp	thlizz	thlishp
sleb	slemth	vleb	vlemth
smul	smutp	hrul	hrutp
swod	swokp	shlod	shlokp
gloob	gloosf	dloob	dloosf
brack	brafsh	vlack	vlafsh
plen	plemth	zmen	zmemth
striff	strishk	shpiff	shpishk
blom	blodb	thlom	thlodb
fleg	flepk	thmeg	thmepk
spood	spooshp	dlood	dlooshp
drass	drathf	thnass	thnathf
shrell	shrethk	shnell	shnethk
trug	trusf	lrug	lrusf
splock	splokp	shlock	shlokp
stip	stithp	shtip	shtithp
froom	frootp	hroom	hrootp

Table B.4. Filler items in Experiment 2.

High acceptability	Low acceptability
bressic	lbafthrizk
claffer	kflisgwevr
spellion	psafzotrl
frotion	tmuhrizb
merly	zgdokpevf
dellous	wlratlumr

APPENDIX C

EXPERIMENTS 3 AND 4 CODE

C.1   Experiment 3 Code

```
from speriment import *
import glob, itertools, random


##### Wording #####


keyboard_instructions = '''For these questions, you will not be able
to use your mouse. Use the F key to choose the option on the left, the
J key to choose the option on the right, and the spacebar to move
forward.'''


exposure_introduction = '''In this part of the experiment, you will
learn words from a made-up language.
<br><br>
Sometimes you'll see one word, and sometimes you'll see two: the
singular version of the word first, and then the plural version.
<br><br>
To help yourself catch on to this new language, type the words you see
into the text box. We recommend pronouncing them out loud, too. In
this language, "ng" is always pronounced as in "singer", never as in
"finger".'''
```

```
exposure_question = '''Please write this word in the text box and
pronounce it to yourself: <br><br>'''


exposure_question_2 = '''Please write these words in the text box and
pronounce them to yourself: <br><br>'''


training_introduction = '''Now we'll focus on learning singulars and
plurals a bit more. You'll see the singular version of a word and two
possible plurals.  Choose the one you think is correct for this
language. Sometimes you'll then be given the correct answer
afterwards, to help you learn. Other times you won't see how you did.
<br><br>
The length of this phase depends on your accuracy. We've found that
people who take their time on these questions actually finish this
phase about <b>much faster</b> than those who rush through the
questions. Take your time and pronounce the correct answers out loud
when they're given to help you learn how this language sounds.'''


catch_intro = '''In this phase, you'll see the singular version of a
word and two possible plurals. Choose the one you think is correct for
this language.'''


test_introduction = '''In this part of the experiment, you will see
pairs of new words.  Based on how each word sounds, decide which one
is more likely to belong to the language you just learned.'''
```

```
test_question = '''Which of these two words is more likely to be a
word of the language you just learned?'''


training_question = '''Which of these words do you think is the
correct plural for this word? <br><br>'''


mouse_instructions = '''Now you can use your mouse again.'''


native_language = "What is your native language?"


daily_language = "What language do you speak at home?"


age = "How old are you?"


sex = "What is your sex?"


strategy = '''What information or strategies did you use to choose
between words in the last section? What did you use in the
singular-plural section?'''


goodbye = '''Thank you for participating in this experiment! Remember
to click Complete HIT on the next page.'''



##### Materials #####


all_pictures = glob.glob('static/images/stim/*.png')
```

```python
plural_pictures = set(glob.glob('static/images/stim/*-pl.png'))
singular_pictures = sorted([pic for pic in all_pictures
    if pic not in plural_pictures])
plural_pictures = sorted(list(plural_pictures))
pictures = [{'singular': sing, 'plural': pl}
    for (sing, pl) in zip(singular_pictures, plural_pictures)]


rows = get_dicts('items_apr28.csv')
item_types = dict([(key, list(val))
    for (key, val)
    in itertools.groupby(rows, lambda r: r['language'])])


##### rows ######
common = item_types['both'] # s and non-alternating pl
train1 = item_types['voice'] # s and pl in -p/-t
train2 = item_types['nasal'] # s and pl in -m
plural = item_types['plural'] # pl in -m and pl in -p/-t


test = item_types['test']
filler = item_types['filler']
#for catch, singular is non-alternating, plural is alternating
voice_catch = item_types['pilot_voice']
nasal_catch = item_types['pilot_nasal']
filler_catch = item_types['pilot_filler']



##### wordbanks ######
```

```python
words1 = [{'singular': row['singular'], 'plural': row['plural']}
          for row in train1]
words2 = [{'singular': row['singular'], 'plural': row['plural']}
          for row in train2]
common_words = [{'singular': row['singular'], 'plural': row['plural']}
                for row in common]
plural_words = [{'m': row['singular'], 'pt': row['plural']}
                for row in plural]


##### Structure #####


with make_experiment(IDGenerator()):


    ###### Exposure Phase #######


    def singular_exposure(rows, wordbank):
        return [Page(
            [exposure_question,
             SampleFrom(wordbank,
                        variable = row['stem'],
                        field = 'singular')],
            options = [Option()],
            freetext = True,
            condition = row['language'],
            keyboard = False,
```

```python
            tags = {'PageType': 'Exposure',
                    'StimulusType': 'Singular'})
         for row in rows]


def plural_exposure(rows, wordbank, lang):
    return [Page(
        [exposure_question,
         SampleFrom(wordbank,
                    variable = row['stem'],
                    field = lang)],
        options = [Option()],
        freetext = True,
        condition = row['language'],
        keyboard = False,
        tags = {'PageType': 'Exposure',
                'StimulusType': 'Plural'})
      for row in rows]


def dual_exposure(rows, wordbank):
    return [Page(
        [exposure_question_2,
            SampleFrom(wordbank,
                       variable = row['stem'],
                       field = 'singular'),
            ' - ',
            SampleFrom(wordbank,
                       variable = row['stem'],
```

```python
                        field = 'plural')],
        options = [Option()],
        freetext = True,
        condition = row['language'],
        keyboard = False,
        tags = {'PageType': 'Exposure',
                'StimulusType': 'Singular-Plural'})
    for row in rows]


### Pages ###
singular_exposure_L1 = singular_exposure(train1, 'words1')
singular_exposure_L2 = singular_exposure(train2, 'words2')
plural_exposure_m = plural_exposure(plural, 'plural_words', 'm')
plural_exposure_pt = plural_exposure(plural, 'plural_words', 'pt')
dual_exposure_L1 = dual_exposure(train1, 'words1')
dual_exposure_L2 = dual_exposure(train2, 'words2')
dual_exposure_both = dual_exposure(common, 'common_words')


### Blocks ###
L1_exposure = Block(pages = singular_exposure_L2 +
    plural_exposure_m + dual_exposure_both + dual_exposure_L1)
L2_exposure = Block(pages = singular_exposure_L1 +
    plural_exposure_pt + dual_exposure_both + dual_exposure_L2)



###### Training Phase ######
```

```python
def alternating_training(rows):
    return [Page(
        [training_question, row['stem']],
        options = [Option(row['singular'],
                            tags = {'Alternates': 'False',
                                    'Segment': row['segment1']},
                            correct = False,
                            feedback = Page([
                                'No, the correct pairing is
                                <br><br>', row['stem'], ' - ',
                                row['plural']],
                                tags = {'PageType': 'Feedback'})),
                   Option(row['plural'],
                            tags = {'Alternates': 'True',
                                    'Segment': row['segment1']},
                            correct = True,
                            feedback = Page(['Correct! <br><br>',
                                row['stem'], ' - ', row['plural']],
                                tags = {'PageType': 'Feedback'}))],
                            keyboard = True,
                            tags = {'PageType': 'Wug',
                                    'ShouldAlternate': True})
        for row in rows]


def non_alternating_training(rows):
    return [Page(
        [training_question, row['stem']],
```

```python
        options = [Option(row['singular'],
                          tags = {'Alternates': 'False',
                                  'Segment': row['segment1']},
                          correct = True,
                          feedback = Page(['Correct! <br><br>',
                                          row['stem'],
                                          ' - ',
                                          row['singular']],
                                  tags = {'PageType': 'Feedback'})),
                  Option(row['plural'],
                          tags = {'Alternates': 'True',
                                  'Segment': row['segment1']},
                          correct = False,
                          feedback = Page([
                              'No, the correct pairing is
                              <br><br>', row['stem'], ' - ',
                              row['singular']],
                              tags = {'PageType': 'Feedback'}))],
                          keyboard = True,
                          tags = {'PageType': 'Wug',
                                  'ShouldAlternate': False})
        for row in rows]


def make_wug_exposure(rows):
    return [Page([training_question, row['stem']],
                options = [Option(row['singular'],
                                  tags = {'Alternates': 'False',
```

```
                                     'Segment': row['segment1']}),
                         Option(row['plural'],
                             tags = {'Alternates': 'True',
                                 'Segment': row['segment1']})],
                 keyboard = True,
                 tags = {"PageType": "Catch"})
             for row in rows]


### Pages ###

filler_training = non_alternating_training(filler_catch)

L1_training_pages = alternating_training(voice_catch) +
    filler_training + make_wug_exposure(nasal_catch)

L2_training_pages = alternating_training(nasal_catch) +
    filler_training + make_wug_exposure(voice_catch)


### Blocks ###

L1_training = Block(pages = L1_training_pages,
                    criterion = 0.8,
                    cutoff = 5)

L2_training = Block(pages = L2_training_pages,
                    criterion = 0.8,
                    cutoff = 5)


###### Testing Phase ######

test_pages_voice = [Page(
        test_question,
```

```python
        options = [
            # b or d
            Option(row['singular'], tags = {'Violation': 'True'}),
            # p or t
            Option(row['voice_satisfying'],
                    tags = {'Violation': 'False'})],
        keyboard = True,
        condition = 'voice',
        tags = {"PageType": 'Test',
                "Violates": "Voicing",
                "Segment": row['segment1']})
    for row in test]


test_pages_place = [Page(
        test_question,
        options = [
            # n or ng
            Option(row['plural'], tags = {'Violation': 'True'}),
            # m
            Option(row['place_satisfying'],
                    tags = {'Violation': 'False'})],
        keyboard = True,
        condition = 'place',
        tags = {"PageType": 'Test',
                "Violates": "Place",
                "Segment": row['segment2']})
    for row in test]
```

```python
filler_pages_v = [Page(
        test_question,
        options = [
            Option(row['singular']), #l
            Option(row['plural'])], # s
        keyboard = True,
        # condition for pseudorandomization
        condition = 'voice',
        tags = {'PageType': 'Filler'})
    for row in filler[:len(filler)/2]]


filler_pages_p = [Page(
        test_question,
        options = [
            Option(row['singular']), #l
            Option(row['plural'])], # s
        keyboard = True,
        # condition for pseudorandomization
        condition = 'place',
        tags = {'PageType': 'Filler'})
    for row in filler[len(filler)/2:]]


test_block = Block(pages = test_pages_voice +
    test_pages_place + filler_pages_v + filler_pages_p,
    pseudorandom = True)
```

```
###### Test if they learned rules #####


precatch = Block(pages = [Page(catch_intro)])


##### Niceties #####


exposure_intro = Block(pages = [Page(exposure_introduction)])

train_intro = Block(pages = [Page(training_introduction)])

test_intro = Block(pages = [Page(test_introduction)])

outro_block = Block(pages = [Page(goodbye)])

keyboard_block = Block(pages = [Page(keyboard_instructions)])

mouse_block = Block(pages = [Page(mouse_instructions)])

demographics_block = Block(pages = [

    Page(age,

        options = [Option(i) for i in range(120)],

        ordered = True),

    Page(sex,

        options = [Option('male'),

                   Option('female'),

                   Option('other')]),

    Page(native_language,

        options = [Option()],

        freetext = True),

    Page(daily_language,

        options = [Option()],

        freetext = True),

    Page(strategy,
```

```
            options = [Option()],

            freetext = True)])


##### Experiment #####

experiment = Experiment([
    # Exposure phase
    exposure_intro,

    L1_exposure, L2_exposure,


        # instructions
        keyboard_block,


        # Training phase
        train_intro,

        L1_training, L2_training,


        # Testing phase
        test_intro,

        test_block,


        # instructions
        mouse_block,


        # finishing up
        demographics_block,

        outro_block],
```

```
        # choose language per participant

        treatments = [[L1_exposure, L1_training],

                      [L2_exposure, L2_training]],

        banks = {'words1': words1,

                 'words2': words2,

                 'common_words': common_words,

                 'plural_words': plural_words,

                 'pictures': pictures})


    experiment.install('alternations_experiment')
```

C.2   Experiment 4 Code

```
from speriment import *

import glob, itertools, random


##### Wording #####


keyboard_instructions = '''For these questions, you will not be able

to use your mouse. Use the F key to choose the option on the left, the

J key to choose the option on the right, and the spacebar to move

forward.'''


exposure_introduction = '''In this part of the experiment, you will

learn words from a made-up language.

<br><br>

Sometimes you'll see one word, and sometimes you'll see two: the
```

singular version of the word first, and then the plural version.

To help yourself catch on to this new language, type the words you see
into the text box. We recommend pronouncing them out loud, too. In
this language, "ng" is always pronounced as in "singer", never as in
"finger".'''

exposure_question = ''''Please write this word in the text box and
pronounce it to yourself:

'''

exposure_question_2 = ''''Please write these words in the text box and
pronounce them to yourself:

'''

training_introduction = ''''Now we'll focus on learning singulars and
plurals a bit more. You'll see the singular version of a word and two
possible plurals. Choose the one you think is correct for this
language.'''

catch_intro = '''In this phase, you'll see the singular version of a
word and two possible plurals. Choose the one you think is correct for
this language.'''

test_introduction = '''In this part of the experiment, you will see
pairs of new words. Unlike the plurals you've seen, these words don't
have any suffixes added. Based on how each word sounds, decide which

one is more likely to belong to the language you just learned.'''

test_question = '''Which of these two words is more likely to be a word of the language you just learned?'''

training_question = '''Which of these words do you think is the correct plural for this word?

'''

mouse_instructions = '''Now you can use your mouse again.'''

native_language = "What is your native language?"

daily_language = "What language do you speak at home?"

age = "How old are you?"

sex = "What is your sex?"

strategy = '''What information or strategies did you use to choose between words in the last section?'''

strategy2 = '''What information or strategies did you use to choose the right plural in the middle of the experiment?'''

goodbye = '''Thank you for participating in this experiment! Remember to click Complete HIT on the next page.'''

```
##### Materials #####

all_pictures = glob.glob('static/images/stim/*.png')

plural_pictures = set(glob.glob('static/images/stim/*-pl.png'))

singular_pictures = sorted([pic for pic in all_pictures if pic not in

plural_pictures]) plural_pictures = sorted(list(plural_pictures))

pictures = [{'singular': sing, 'plural': pl} for (sing, pl) in

        zip(singular_pictures, plural_pictures)]


rows = get_dicts('items_apr28.csv')

item_types = dict([(key, list(val)) for (key, val)

    in itertools.groupby(rows, lambda r: r['language'])])


##### rows ######
#5
common = item_types['both'] # s and non-alternating pl
#10
train1 = item_types['voice'] # s and pl in -p/-t
#10
train2 = item_types['nasal'] # s and pl in -m
#10
plural = item_types['plural'] # pl in -m and pl in -p/-t
# L1 should show common, train1, train2 singular, "plural" singular
# L2 should show common, train2, train1 singular, "plural" plural


test = item_types['test']
```

```python
filler = item_types['filler']

#for catch, "singular" is non-alternating, "plural" is alternating

voice_catch = item_types['pilot_voice']

nasal_catch = item_types['pilot_nasal']

filler_catch = item_types['pilot_filler']




##### wordbanks ######


words1 = [{'singular': row['singular'], 'plural': row['plural']}
          for row in train1]

words2 = [{'singular': row['singular'], 'plural': row['plural']}
          for row in train2]

common_words = [{'singular': row['singular'], 'plural': row['plural']}
                for row in common]

plural_words = [{'m': row['singular'], 'pt': row['plural']}
                for row in plural]


##### Structure #####


with make_experiment(IDGenerator()):


    ###### Exposure Phase #######


    def singular_exposure(rows, wordbank):
        return [Page([exposure_question,
```

```python
                        SampleFrom(wordbank,
                                   variable = row['stem'],
                                   field = 'singular')],
                options = [Option()],
                freetext = True,
                condition = row['language'],
                keyboard = False,
                tags = {'PageType': 'Exposure',
                        'StimulusType': 'Singular'})
            for row in rows]


def plural_exposure(rows, wordbank, lang):
    return [Page([exposure_question,
                  SampleFrom(wordbank,
                             variable = row['stem'],
                             field = lang)],
                options = [Option()],
                freetext = True,
                condition = row['language'],
                keyboard = False,
                tags = {'PageType': 'Exposure',
                        'StimulusType': 'Plural'})
            for row in rows]


def dual_exposure(rows, wordbank):
    return [Page([exposure_question_2,
                  SampleFrom(wordbank,
```

```
                              variable = row['stem'],

                              field = 'singular'),

                    ' - ',

                   SampleFrom(wordbank,

                              variable = row['stem'],

                              field = 'plural')],

              options = [Option()],

              freetext = True,

              condition = row['language'],

              keyboard = False,

              tags = {'PageType': 'Exposure',

                      'StimulusType': 'Singular-Plural'})

         for row in rows]


### Pages ###

singular_exposure_L1 = singular_exposure(train1, 'words1')

singular_exposure_L2 = singular_exposure(train2, 'words2')

plural_exposure_m = plural_exposure(plural, 'plural_words', 'm')

plural_exposure_pt = plural_exposure(plural, 'plural_words', 'pt')

dual_exposure_L1 = dual_exposure(train1, 'words1')

dual_exposure_L2 = dual_exposure(train2, 'words2')

dual_exposure_both = dual_exposure(common, 'common_words')


### Blocks ###

L1_exposure = Block(pages = singular_exposure_L2 +

    plural_exposure_m + dual_exposure_both + dual_exposure_L1)

L2_exposure = Block(pages = singular_exposure_L1 +
```

```
                  plural_exposure_pt + dual_exposure_both + dual_exposure_L2)



###### Training Phase ######


def alternating_training(rows):
    return [Page([training_question, row['stem']],
                options = [
                    Option(row['singular'],
                            tags = {'Alternates': 'False',
                                    'Segment': row['segment1']},
                            correct = False),
                    Option(row['plural'],
                            tags = {'Alternates': 'True',
                                    'Segment': row['segment1']},
                            correct = True)],
                keyboard = True,
                tags = {'PageType': 'Wug',
                        'ShouldAlternate': True})
            for row in rows]


def non_alternating_training(rows):
    return [Page([training_question, row['stem']],
                options = [
                    Option(row['singular'],
                            tags = {'Alternates': 'False',
                                    'Segment': row['segment1']},
```

```
                                            correct = True),
                               Option(row['plural'],
                                           tags = {'Alternates': 'True',
                                                      'Segment': row['segment1']},
                                           correct = False)],
                        keyboard = True,
                        tags = {'PageType': 'Wug',
                                   'ShouldAlternate': False})
              for row in rows]


def make_wug_exposure(rows):
    return [Page([training_question, row['stem']],
                 options = [
                       Option(row['singular'],
                                   tags = {'Alternates': 'False',
                                              'Segment': row['segment1']}),
                       Option(row['plural'],
                                   tags = {'Alternates': 'True',
                                              'Segment': row['segment1']})],
                 keyboard = True,
                 tags = {"PageType": "Catch"})
              for row in rows]


### Pages ###
filler_training = non_alternating_training(filler_catch)
L1_training_pages = alternating_training(voice_catch) +
    filler_training + make_wug_exposure(nasal_catch)
```

```
L2_training_pages = alternating_training(nasal_catch) +

    filler_training + make_wug_exposure(voice_catch)


### Blocks ###

L1_training = Block(pages = L1_training_pages)

L2_training = Block(pages = L2_training_pages)


###### Testing Phase ######


test_pages_voice = [Page(

        test_question,

        options = [

            # b or d

            Option(

                row['singular'],

                tags = {'Violation': 'True'}),

            # p or t

            Option(

                row['voice_satisfying'],

                tags = {'Violation': 'False'})],

        keyboard = True,

        condition = 'voice',

        tags = {"PageType": 'Test',

                "Violates": "Voicing",

                "Segment": row['segment1']})

    for row in test]
```

```python
test_pages_place = [Page(
        test_question,
        options = [
            # n or ng
            Option(
                row['plural'],
                tags = {'Violation': 'True'}),
            # m
            Option(
                row['place_satisfying'],
                tags = {'Violation': 'False'})],
        keyboard = True,
        condition = 'place',
        tags = {"PageType": 'Test',
                "Violates": "Place",
                "Segment": row['segment2']})
    for row in test]

filler_pages_v = [Page(
        test_question,
        options = [
            Option(row['singular']), # l
            Option(row['plural'])], # s
        keyboard = True,
        # condition included for pseudorandomization
        condition = 'voice',
        tags = {'PageType': 'Filler'})
```

```
        for row in filler[:len(filler)/2]]


filler_pages_p = [Page(
        test_question,
        options = [
            Option(row['singular']), # l
            Option(row['plural'])], # s
        keyboard = True,
        # condition included for pseudorandomization
        condition = 'place',
        tags = {'PageType': 'Filler'})
    for row in filler[len(filler)/2:]]


test_block = Block(pages = test_pages_voice + test_pages_place +
    filler_pages_v + filler_pages_p, pseudorandom = True)


###### Test if they learned rules #####


precatch = Block(pages = [Page(catch_intro)])


##### Niceties #####


exposure_intro = Block(pages = [Page(exposure_introduction)])
train_intro = Block(pages = [Page(training_introduction)])
test_intro = Block(pages = [Page(test_introduction)])
outro_block = Block(pages = [Page(goodbye)])
keyboard_block = Block(pages = [Page(keyboard_instructions)])
```

```python
mouse_block = Block(pages = [Page(mouse_instructions)])
demographics_block = Block(pages = [

    Page(age,

        options = [Option(i) for i in range(120)],

        ordered = True),

    Page(sex,

        options = [Option('male'),

                    Option('female'),

                    Option('other')]),

    Page(native_language,

        options = [Option()],

        freetext = True),

    Page(daily_language,

        options = [Option()],

        freetext = True),

    Page(strategy,

        options = [Option()],

        freetext = True),

    Page(strategy2,

        options = [Option()],

        freetext = True)])


##### Experiment #####


experiment = Experiment([

    # exposure phase

    exposure_intro,
```

```
        L1_exposure, L2_exposure,


        # instructions
        keyboard_block,


        # training phase
        train_intro,
        L1_training, L2_training,


        # test phase
        test_intro,
        test_block,


        # instructions
        mouse_block,


        # finishing up
        demographics_block,
        outro_block],


    # choose one language per participant
    treatments = [[L1_exposure, L1_training],
        [L2_exposure, L2_training]],
    banks = {'words1': words1,
            'words2': words2,
            'common_words': common_words,
            'plural_words': plural_words,
```

```
                  'pictures': pictures})


experiment.install('alternations_experiment')
```

APPENDIX D

EXPERIMENTS 3 AND 4 STIMULI

Table D.1. Treatment-neutral exposure items in Experiments 3 and 4.

Singular	Plural
teldus	teldusfa
labap	labapfa
fongem	fongemfa
pufsit	pufsitfa
nimol	nimolfa

Table D.2. Devoicing treatment exposure items appearing in both the singular and plural in Experiments 3 and 4.

Singular	Plural
nemab	nemapfa
sobud	sobutfa
fetab	fetapfa
tamdid	tamditfa
dungib	dungipfa
milod	milotfa
lispeb	lispepfa
sangod	sangotfa
poneb	ponepfa
bupfud	bupfutfa

Table D.3. Devoicing treatment exposure items appearing in the singular only in Experiments 3 and 4.

Singular Only
sumen
lobon
pitang
demding
fangun
solon
buspang
nangung
tinen
mepfing

Table D.4. Devoicing treatment exposure items appearing in the plural only in Experiments 3 and 4.

Plural Only
funemfa
semomfa
nangimfa
sipumfa
lotamfa
melomfa
bangumfa
tubemfa
ditfamfa
podimfa

Table D.5. Place Assimilation treatment exposure items appearing in both the singular and plural in Experiments 3 and 4.

Singular	Plural
sumen	sumemfa
lobon	lobomfa
pitang	pitamfa
demding	demdimfa
fangun	fangumfa
solon	solomfa
buspang	buspamfa
nangung	nangumfa
tinen	tinemfa
mepfing	mepfimfa

Table D.6. Place Assimilation treatment exposure items appearing in the singular only in Experiments 3 and 4.

Singular Only
nemab
sobud
fetab
tamdid
dungib
milod
lispeb
sangod
poneb
bupfud

Table D.7. Place Assimilation treatment exposure items appearing in the plural only in Experiments 3 and 4.

Plural Only
funepfa
semotfa
nangipfa
siputfa
lotapfa
melotfa
bangupfa
tubetfa
ditfapfa
poditfa

Table D.8. Treatment-neutral training items in Experiments 3 and 4.

Singular	Plural 1	Plural 2
beful	befulfa	befumfa
pidep	pidepfa	pidetfa
nupis	nupisfa	nupiffa
tatul	tatulfa	tatumfa
dulim	dulimfa	dulilfa
fongep	fongepfa	fongetfa
mengos	mengosfa	mengoffa
limat	limatfa	limapfa

Table D.9. Devoicing treatment training items in Experiments 3 and 4.

Singular	Plural 1	Plural 2
sapod	sapodfa	sapotfa
pulob	pulobfa	pulopfa
lumub	lumubfa	lumupfa
tetib	tetibfa	tetipfa
fadud	fadudfa	fadutfa
mofeb	mofebfa	mofepfa
bined	binedfa	binetfa
ningad	ningadfa	ningatfa
sengib	sengibfa	sengipfa
dobad	dobadfa	dobatfa

Table D.10. Place Assimilation treatment training items in Experiments 3 and 4.

Singular	Plural 1	Plural 2
sapong	sapongfa	sapomfa
pulong	pulongfa	pulomfa
lumung	lumungfa	lumumfa
tetin	tetinfa	tetimfa
fadung	fadungfa	fadumfa
mofeng	mofengfa	mofemfa
binen	binenfa	binemfa
ningan	ninganfa	ningamfa
sengin	senginfa	sengimfa
doban	dobanfa	dobamfa
sanot	sanotfa	sanopfa
sobam	sobamfa	sobalfa

Table D.11. Filler items in the testing phase of Experiments 3 and 4.

Form 1	Form 2
folfam	fosfam
mulfuf	musfuf
lalfit	lasfit
bilfab	bisfab
lelfeng	lesfeng
sulfen	susfen
nilfid	nisfid
polfup	posfup
delfong	desfong
talfos	tasfos

Table D.12. Items testing *DF in Experiments 3 and 4.

Constraint Violating Form	Constraint Satisfying Form
ludfum	lutfum
pidfap	pitfap
todfob	totfob
budfil	butfil
didfing	ditfing
sudfuf	sutfuf
nadfung	natfung
madfas	matfas
medfet	metfet
nidfit	nitfit
nabfep	napfep
tibfim	tipfim
fobfol	fopfol
pebfuf	pepfuf
bubfan	bupfan
sabfod	sapfod
fobfes	fopfes
bobfang	bopfang
lebfong	lepfong
debfed	depfed

Table D.13. Items testing *NF in Experiments 3 and 4.

Constraint Violating Form	Constraint Satisfying Form
lunfum	lumfum
pingfap	pimfap
tonfob	tomfob
bungfil	bumfil
dinfing	dimfing
sungfuf	sumfuf
nanfung	namfung
mangfas	mamfas
menfet	memfet
ningfit	nimfit
nanfep	namfep
tingfim	timfim
fonfol	fomfol
pengfuf	pemfuf
bunfan	bumfan
sangfod	samfod
fonfes	fomfes
bongfang	bomfang
lenfong	lemfong
dengfed	demfed

BIBLIOGRAPHY

Adriaans, F. and Kager, R. (2010). Adding generalization to statistical learning: The induction of phonotactics from continuous speech. *Journal of Memory and Language*, 62(3):311–331.

Agile Alliance (2001). Agile manifesto. http://www.agilemanifesto.org.

Albright, A. (2008a). From clusters to words: Grammatical models of nonce word acceptability. Handout of talk presented at 82nd LSA, Chicago. http://web.mit.edu/albright/www/papers/Albright-LSA2008Handout.pdf.

Albright, A. (2008b). Gradient phonological acceptability as a grammatical effect. http://web.mit.edu/albright/www/papers/Albright-GrammaticalGradience.pdf.

Albright, A. (2009a). Cumulative violations and complexity thresholds: Evidence from Lakhota. In *17th Manchester Phonology Meeting*.

Albright, A. (2009b). Feature-based generalisation as a source of gradient acceptability. *Phonology*, 26(1):9–41.

Anttila, A. (2009). Derived environment effects in colloquial Helsinki Finnish. *The nature of the word: essays in honor of Paul Kiparsky*, pages 433–460.

Armitage, P., McPherson, C. K., and Rowe, B. C. (1969). Repeated Significance Tests on Accumulating Data. *Journal of the Royal Statistical Society. Series A (General)*, 132(2):235.

Bailey, T. and Hahn, U. (2001). Determinants of wordlikeness: Phonotactics or lexical neighborhoods? *Journal of Memory and Language*, 44(4):568–591.

Bates, D. and Maechler, M. (2009). *lme4: Linear mixed-effects models using S4 classes*. R package version 0.999375-31.

Becker, M. and Levine, J. (2010). *Experigen — an online experiment platform*. Available at https://github.com/tlozoot/experigen.

Becker, M., Nevins, A., and Ketrez, N. (2011). The surfeit of the stimulus: Analytic biases filter lexical statistics in Turkish laryngeal alternations. *Language*, 87(1):84–125.

Berent, I., Everett, D., and Shimron, J. (2001). Do phonological representations specify formal variables? Evidence from the Obligatory Contour Principle. *Cognitive Psychology*, 42:1–60.

Berent, I., Lennertz, T., Smolensky, P., and Vaknin-Nusbaum, V. (2009). Listeners' knowledge of phonological universals: evidence from nasal clusters. *Phonology*, 26(01):75–108.

Berent, I., Steriade, D., Lennertz, T., and Vaknin, V. (2007). What we know about what we have never heard: Evidence from perceptual illusions. *Cognition*, 104(3):591–630.

Boersma, P. and Pater, J. (2008). Convergence properties of a gradual learning algorithm for Harmonic Grammar. ROA 970.

Breen, M., Kingston, J., and Sanders, L. D. (2013). Perceptual representations of phonotactically illegal syllables. *Attention, Perception, & Psychophysics*, 75(1):101–120.

Brown, R. and Hildum, D. (1956). Expectancy and the perception of syllables. *Language*, 32(3):411–419.

Chomsky, N. and Halle, M. (1968). *The Sound Pattern of English*. Harper and Row, New York.

Coetzee, A. (2004). *What It Means to Be a Loser: Non-optimal Candidates in Optimality Theory*. Doctoral dissertation, University of Massachusetts Amherst.

Coetzee, A. W. and Pater, J. (2008). Weighted constraints and gradient restrictions on place co-occurrence in Muna and Arabic. *Natural Language & Linguistic Theory*, 26(2):289–337.

Coleman, J. and Pierrehumbert, J. (1997). Stochastic phonological grammars and acceptability. In *Computational Phonology: Third meeting of the ACL special interest group in computational phonology*, pages 49–56.

Crump, M. J. C., McDonnell, J. V., and Gureckis, T. M. (2013). Evaluating Amazon's Mechanical Turk as a Tool for Experimental Behavioral Research. *PLoS ONE*, 8(3):e57410.

Daland, R., Hayes, B., White, J., Garellek, M., Davis, A., and Norrmann, I. (2011). Explaining sonority projection effects. *Phonology*, 28(2):197–234.

Darcy, I., Ramus, F., Christophe, A., Kinzler, K., and Dupoux, E. (2009). Phonological knowledge in compensation for native and non-native assimilation. *Variation and gradience in phonetics and phonology*, 14:265.

Drummond, A. (2011). IbexFarm. Available at spellout.net/ibexfarm.

Dupoux, E., Kakehi, K., Hirose, Y., Pallier, C., and Mehler, J. (1999). Epenthetic vowels in Japanese: A perceptual illusion? *Journal of Experimental Psychology Human Perception and Performance*, 25(6):1568–1578.

Erlewine, M. Y. and Kotek, H. (2013). A streamlined approach to online linguistic surveys. http://ling.auf.net/lingbuzz/001802/, manuscript, Massachusetts Institute of Technology.

Friederici, A. D. and Wessels, J. M. (1993). Phonotactic knowledge of word boundaries and its use in infant speech perception. *Perception & psychophysics*, 54(3):287–295.

Frisch, S. A., Large, N. R., and Pisoni, D. B. (2000). Perception of wordlikeness: Effects of segment probability and length on the processing of nonwords. *Journal of memory and language*, 42(4):481–496.

Gibson, E. and Fedorenko, E. (2013). The need for quantitative methods in syntax and semantics research. *Language and Cognitive Processes*, 28(1-2):88–124.

Goldwater, S. and Johnson, M. (2003). Learning OT constraint rankings using a Maximum Entropy model. In Spenader, J., Eriksson, A., and Östen Dahl, editors, *Proceedings of the Workshop on Variation within Optimality theory*, pages 111–120. Stockholm University.

Green, C. R. and Davis, S. (2014). Superadditivity and limitations on syllable complexity in Bambara words. In Farris-Trimble, A. W. and Barlow, J. A., editors, *Perspectives on phonological theory and development, in honor of Daniel A. Dinnsen*, pages 223–47.

Hallé, P. A. and Best, C. T. (2007). Dental-to-velar perceptual assimilation: A cross-linguistic study of the perception of dental stop+/l/clusters. *The Journal of the Acoustical Society of America*, 121(5):2899–2914.

Hallé, P. A., Segui, J., Frauenfelder, U., and Meunier, C. (1998). Processing of illegal consonant clusters: A case of perceptual assimilation? *Journal of experimental psychology: Human perception and performance*, 24(2):592–608.

Hammond, M. (1999). *The Phonology of English: A Prosodic Optimality-Theoretic Approach: A Prosodic Optimality-Theoretic Approach*. Oxford University Press.

Hayes, B. (2004). Phonological acquisition in Optimality Theory: The early stages. In Kager, R., Pater, J., and Zonneveld, W., editors, *Constraints in Phonological Acquisition*. Cambridge University Press, Cambridge.

Hayes, B. and Wilson, C. (2008). A Maximum Entropy model of phonotactics and phonotactic learning. *Linguistic Inquiry*, 39(3):379–440.

Hayes, B., Zuraw, K., Siptár, P., and Londe, Z. C. (2009). Natural and unnatural constraints in Hungarian. *Language*, 85:822–863.

Jakobson, R. (1978). Mutual Assimilation of Russian Voiced and Voiceless Consonants. *Studia Linguistica*, 32(1-2):107–110.

Jarosz, G. (2006). *Rich Lexicons and Restrictive Grammars – Maximum Likelihood Learning in Optimality Theory.* Doctoral dissertation, Johns Hopkins University.

John, L. K., Loewenstein, G., and Prelec, D. (2012). Measuring the prevalence of questionable research practices with incentives for truth telling. *Psychological science*, 23:524–532.

Jun, J. (2004). Place assimilation. In Hayes, B., Kirchner, R., and Steriade, D., editors, *Phonetically Based Phonology*, pages 58–86. Cambridge University Press, Cambridge.

Jusczyk, P. W., Friederici, A. D., Wessels, J. M., Svenkerud, V. Y., and Jusczyk, A. M. (1993). Infants' sensitivity to the sound patterns of native language words. *Journal of memory and language*, 32(3):402–420.

Jusczyk, P. W. and Luce, P. A. (1994). Infants' sensitivity to phonotactic patterns in the native language. *Journal of Memory and Language*, 33(5):630–645.

Keller, F. (2000). *Gradience in Grammar: Experimental and Computational Aspects of Degrees of Grammaticality.* PhD thesis, University of Edinburgh.

Keller, F. (2006). Linear Optimality Theory as a model of gradience in grammar. In Fanselow, G., Féry, C., Vogel, R., and Schelesewsky, M., editors, *Gradience in Grammar: Generative Perspectives*, pages 270–288. Oxford University Press, Oxford.

Keller, F., Gunasekharan, S., Mayo, N., and Corley, M. (2009a). Timing accuracy of web experiments: A case study using the WebExp software package. *Behavior Research Methods*, 41(1):1–12.

Keller, F., Gunasekharan, S., Mayo, N., and Corley, M. (2009b). Timing accuracy of web experiments: A case study using the WebExp software package. *Behavior Research Methods*, 41(1):1–12. Available at http://link.springer.com/article/10.3758/BRM.41.1.12.

Kiparsky, P. (1973). Phonological representations. In Fujimura, O. and Fujimura, O., editors, *Three Dimensions of Linguistic Theory*, pages 3–136. TEC, Tokyo.

Kiparsky, P. (1985). Some consequences of Lexical Phonology. *Phonology*, 2:85–138.

Kisseberth, C. (1970). On the functional unity of phonological rules. *Linguistic Inquiry*, 1:291–306.

Legendre, G., Miyata, Y., and Smolensky, P. (1990). Harmonic Grammar – A formal multi-level connectionist theory of linguistic well-formedness: An Application. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pages 884–891. Lawrence Erlbaum Associates, Mahwah, NJ.

LimeSurvey Project Team / Carsten Schmitz (2012). *LimeSurvey: An Open Source Survey Tool.* Hamburg, Germany. Available at http://www.limesurvey.org.

Lubowicz, A. (2002). Derived Environment Effects in Optimality Theory. *Lingua*, 112:243–280.

Luce, R. D. (1959). *Individual choice behavior.* Wiley, New York.

Martin, A. (2011). Grammars leak: Modeling how phonotactic generalizations interact within the grammar. *Language*, 87(4):751–770.

Mascaró, J. (1976). *Catalan phonology and the phonological cycle.* PhD thesis, Massachusetts Institute of Technology.

Massaro, D. W. and Cohen, M. M. (1983). Phonological context in speech perception. *Perception & Psychophysics*, 34(4):338–348.

McCarroll, D., Crays, N., and Dunlap, W. P. (1992). Sequential ANOVAs and type I error rates. *Educational and psychological measurement*, 52(2):387–393.

McCarthy, J. J. (2003). Comparative markedness. *Theoretical Linguistics*, 29:1–51.

McDonnell, J., Martin, J., Markant, D., Coenen, A., Rich, A., and Gureckis, T. (2012). *psiTurk (Version 1.02) [Software].* New York, NY. Available at https://github.com/NYUCCL/psiTurk.

Ohala, J. J. and Ohala, M. (1986). Testing hypotheses regarding the psychological manifestation of morpheme structure constraints. In Ohala, J. J. and Jaeger, J. J., editors, *Experimental Phonology*, pages 239–252. Academic Press, Orlando.

Pater, J. (2007). Cumulative ill-formedness in typological and experimental data. Paper presented at the Conference on Experimental Approaches to Optimality Theory, University of Michigan. http://people.umass.edu/pater/pater-michigan.pdf.

Pater, J. (2009). Weighted constraints in generative linguistics. *Cognitive Science*, 33(6):999–1035.

Pater, J. and Tessier, A.-M. (2003). Phonotactic knowledge and the acquisition of alternations. In Solé, M. J., Recasens, D., and Romero, J., editors, *Proceedings of the 15th International Congress of Phonetic Sciences*, pages 1177–1180. Universitat Autònoma de Barcelona, Barcelona.

Phillips, C. (2009). Should we impeach armchair linguists. *Japanese/Korean Linguistics*, 17:49–64.

Pierrehumbert, J. B. (2006). The statistical basis of an unnatural alternation. *Laboratory phonology 8*, pages 81–106.

Pocock, S. J. (1977). Group sequential methods in the design and analysis of clinical trials. *Biometrika*, 64(2):191–199.

Potts, C., Pater, J., Jesney, K., Bhatt, R., and Becker, M. (2010). Harmonic Grammar with Linear Programming: From linear systems to linguistic typology. *Phonology*, 27(1):77–117.

Prince, A. and Smolensky, P. (1993/2004). *Optimality Theory: Constraint Interaction in Generative Grammar*. Malden, MA, and Oxford, UK: Blackwell.

R Development Core Team (2011). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Royce, W. W. (1970). Managing the development of large software systems. In *proceedings of IEEE WESCON*, volume 26, pages 328–388. Los Angeles.

Sagarin, B. J., Ambler, J. K., and Lee, E. M. (2014). An ethical approach to peeking at data. *Perspectives on Psychological Science*, 9(3):293–304.

Sapir, E. and Hoijer, H. (1967). *The phonology and morphology of the Navaho language*, volume 50. University of California Press.

Shields, P. G. (2000). Publication bias is a scientific problem with adverse ethical outcomes: the case for a section for null results. *Cancer Epidemiology Biomarkers & Prevention*, 9(8):771–772.

Shih, S. S. (2015). Super additive phonological similarity as constraint conjunction. *38th Generative Linguisticcs in the Old World (GLOW), University of Paris Diderot.* http://cogsci.ucmerced.edu/shih/Shih_GLOW2015.pdf.

Smolensky, P. (2006). Optimality in phonology II: Harmonic completeness, local constraint conjunction, and feature-domain markedness. In Smolensky, P. and Legendre, G., editors, *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar*, pages 585–720. MIT Press/Bradford Books, Cambridge, MA.

Smolensky, P. and Legendre, G. (2006). *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar*. MIT Press, Cambridge, MA.

Sprouse, J. (2011). A validation of Amazon Mechanical Turk for the collection of acceptability judgments in linguistic theory. *Behavior research methods*, 43(1):155–167.

Sterling, T. D. (1959). Publication decisions and their possible effects on inferences drawn from tests of significance-or vice versa. *Journal of the American statistical association*, 54(285):30–34.

Strube, M. J. (2006). SNOOP: A program for demonstrating the consequences of premature and repeated null hypothesis testing. *Behavior Research Methods*, 38(1):24–27.

SurveyMonkey, Inc. (1999). *SurveyMonkey*. Palo Alto, CA. Available at www.surveymonkey.com.

Tesar, B. and Prince, A. (2004). Using phonotactics to learn phonological alternations. In *CLS 39, Part II: The Panels*. Chicago Linguistic Society, Chicago.

Todd, S., Whitehead, A., Stallard, N., and Whitehead, J. (2001). Interim analyses and sequential designs in phase III studies. *British Journal of Clinical Pharmacology*, 51(5):394–399.

Tosch, E. and Berger, E. D. (2014). Surveyman: Programming and automatically debugging surveys. In *Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications*, pages 197–211. ACM.

Van Oostendorp, M. (2007). Derived environment effects and consistency of exponence. http://www.vanoostendorp.nl/pdf/freedom_oostendorp.pdf.

Vitevitch, M. and Luce, P. (2004). A web-based interface to calculate phonotactic probability for words and nonwords in english. *Behavior Research Methods*, 36(3):481–487.

White, K. S., Peperkamp, S., Kirk, C., and Morgan, J. L. (2008). Rapid acquisition of phonological alternations by infants. *Cognition*, 107(1):238–265.

Wolf, M. (2008). *Optimal interleaving: serial phonology-morphology interaction in a constraint-based model*. Doctoral Dissertation, University of Massachusetts, Amherst, Amherst, Mass.

Young, R. W. and Morgan, W. (1980). *The Navajo language: A grammar and colloquial dictionary*, volume 3. University of New Mexico Press.