Universität des Saarlandes

Connectionist Language Production: Distributed Representations and the Uniform Information Density Hypothesis

Dissertation
zur Erlangung des akademischen Grades eines

Doktors der Philosophie

der Philosophischen Fakultät
der Universität des Saarlandes

vorgelegt von
Jesús Calvillo
aus Morelia, Mexiko

Saarbrücken, 2019

Dekan der Fakultät:	Prof. Dr. Heinrich Schlange-Schöningen
Erstberichterstatter:	Prof. Dr. Matthew W Crocker
Zweitberichterstatter:	Prof. Dr. Stefan Frank
Tag der letzten Prüfungsleistung:	20/05/2019

Acknowledgements

This PhD was a very interesting experience in which I learned plenty, both as a researcher and as a person. During this time I was not alone, I was blessed with many people, who helped me and taught me, and to whom I will always be grateful.

Among those people, I would like to express my gratitude first and foremost to my advisor Matthew W Crocker. He guided me through this PhD while giving me the freedom to explore the ideas that I found most interesting. From him I learned about psycholinguistics, how science works, how to structure and pursue research, among many other lessons that will help me in my life. He is an excellent person and researcher, and I feel very honoured to have him as advisor.

I would also like to acknowledge the help that I received from Harm Brouwer. He provided the code to generate the situation state space that was used throughout this dissertation, as well as comments that helped during the shaping of this work. I am grateful for that.

I am also very grateful to my research group, who was always eager to help and give advice. I was very happy to work with you and I am looking forward to future collaborations and/or drinks.

Not everything is work, and work cannot be everything. There were people who supported me personally in this stage of my life. From them I took strength, energy and comfort that allowed me to pursue my goals. For that, I would like to thank my friends who spent some of their time with me throughout these years. I learned so much from you, and I treasure so many memories that I will always be in debt to you.

My family has always supported me in my projects, even when I am so far away. I am very happy and grateful to know that I can always rely on you.

I am also grateful to Saarbrücken, which is a wonderful crazy city with amazing people, in which I always feel welcomed. It will always be a place close to my heart.

I would also like to thank the Mexican National Council of Science and Technology (CONACYT) for the support that allowed me to pursue this degree.

Finally, I dedicate this work to my father Antonio Calvillo Paz, who taught me that for every problem there is a solution, you just have to keep looking and never give up.

Connectionist Language Production: Distributed Representations and the Uniform Information Density Hypothesis

Abstract

This dissertation approaches the task of modeling human sentence production from a connectionist point of view, and using distributed semantic representations. The main questions it tries to address are: (i) whether the distributed semantic representations defined by Frank et al. (2009) are suitable to model sentence production using artificial neural networks, (ii) the behavior and internal mechanism of a model that uses this representations and recurrent neural networks, and (iii) a mechanistic account of the Uniform Information Density Hypothesis (UID; Jaeger, 2006; Levy and Jaeger, 2007).

Regarding the first point, the semantic representations of Frank et al. (2009), called *situation vectors* are points in a vector space where each vector contains information about the observations in which an event and a corresponding sentence are true. These representations have been successfully used to model language comprehension (e.g., Frank et al., 2009; Venhuizen et al., 2018). During the construction of these vectors, however, a dimensionality reduction process introduces some loss of information, which causes some aspects to be no longer recognizable, reducing the performance of a model that utilizes them. In order to address this issue, *belief vectors* are introduced, which could be regarded as an alternative way to obtain semantic representations of manageable dimensionality. These two types of representations (situation and belief vectors) are evaluated using them as input for a sentence production model that implements an extension of a Simple Recurrent Neural network (Elman, 1990). This model was tested under different conditions corresponding to different levels of *systematicity*, which is the ability of a model to generalize from a set of known items to a set of novel ones. Systematicity is an essential attribute that a model of sentence processing has to possess, considering that the number of sentences that can be generated for a given language is infinite, and therefore it is not feasible to memorize all possible message-sentence pairs. The results showed that the model was able to generalize with a very high performance in all test conditions, demonstrating a systematic behavior. Furthermore, the errors that it elicited were related to very similar semantic representations, reflecting the speech error literature, which states that speech errors involve elements with semantic or phonological similarity. This result further demonstrates the systematic behavior of the model, as it processes similar semantic representations in a similar way, even if they are new to the model.

Regarding the second point, the sentence production model was analyzed in two different ways. First, by looking at the sentences it produces, including the errors elicited, highlighting difficulties and preferences of the model. The results revealed that the model learns the syntactic patterns of the language, reflecting its statistical nature, and that its main difficulty is related to very similar semantic representations, sometimes producing unintended sentences that are however very semantically related to the intended ones. Second, the connection weights and activation patterns of the model were also analyzed, reaching an algorithmic account of the internal processing of the model. According to this, the input semantic representation activates the words that are related to its content, giving an idea of their order by providing relatively more activation to words that are likely to appear early in the sentence. Then, at each time step the word that was previously produced activates syntactic and semantic constraints on the next word productions, while the context units of the recurrence preserve information through time, allowing the model to enforce long distance dependencies. We propose that these results can inform about the internal processing of models with similar architecture.

Regarding the third point, an extension of the model is proposed with the goal of modeling UID. According to UID, language production is an efficient process affected by a tendency to produce linguistic units distributing the information as uniformly as possible and close to the capacity of the communication channel, given the encoding possibilities of the language, thus optimizing the amount of information that is transmitted per time unit. This extension of the model approaches UID by balancing two different production strategies: one where the model produces the word with highest probability given the semantics and the previously produced words, and another one where the model produces the word that would minimize the sentence length given the semantic representation and the previously produced words. By combining these two strategies, the model was able to produce sentences with different levels of information density and uniformity, providing a first step to model UID at the algorithmic level of analysis.

In sum, the results show that the distributed semantic representations of Frank et al. (2009) can be used to model sentence production, exhibiting systematicity. Moreover, an algorithmic account of the internal behavior of the model was reached, with the potential to generalize to other models with similar architecture. Finally, a model of UID is presented, highlighting some important aspects about UID that need to be addressed in order to go from the formulation of UID at the computational level of analysis to a mechanistic account at the algorithmic level.

Kurzzusammenfassung

Diese Dissertation widmet sich der Aufgabe, die menschliche Satzproduktion aus konnektionistischer Sicht zu modellieren und dabei verteilte semantische Repräsentationen zu verwenden. Die Schwerpunkte werden dabei sein: (i) die Frage, ob die von Frank et al. (2009) definierten verteilten semantischen Repräsentationen geeignet sind, um die Satzproduktion unter Verwendung künstlicher neuronaler Netze zu modellieren; (ii) das Verhalten und der interne Mechanismus eines Modells, das diese Repräsentationen und rekurrente neuronale Netze verwendet; (iii) eine mechanistische Darstellung der Uniform Information Density Hypothesis (UID; Jaeger, 2006; Levy and Jaeger, 2007).

Zunächst sei angenommen, dass die Repräsentationen von Frank et al. (2009), genannt *Situation Vektoren*, Punkte in einem Vektorraum sind, in dem jeder Vektor Informationen über Beobachtungen enthält, in denen ein Ereignis und ein entsprechender Satz wahr sind. Diese Repräsentationen wurden erfolgreich verwendet, um Sprachverständnis zu modellieren (z.B. Frank et al., 2009; Venhuizen et al., 2018). Während der Konstruktion dieser Vektoren führt ein Prozess der Dimensionsreduktion jedoch zu einem gewissen Informationsverlust, wodurch einige Aspekte verloren gehen. Um das Problem zu lösen, werden als Alternative *Belief Vektoren* eingeführt. Diese beiden Arten der Repräsentation werden ausgewertet, indem sie als Eingabe für ein Satzproduktionsmodell verwendet werden, welches als Erweiterung eines Simple Recurrent Neural Network (SRN, Elman, 1990) implementiert wurden. Dieses Modell wird unter verschiedenen Bedingungen getestet, die verschiedenen Ebenen der *Systematizität* entsprechen, d.h. der Fähigkeit eines Modells, von einer Menge bekannter Elemente auf eine Menge neuer Elemente zu verallgemeinern. Systematizität ist ein wesentliches Attribut, das ein Modell der Satzverarbeitung besitzen muss, wenn man bedenkt, dass die Anzahl der Sätze, die in einer bestimmte Sprache erzeugt werden können, unendlich ist und es daher nicht möglich ist, sich alle möglichen Nachrichten-Satz-Paare zu merken. Die Ergebnisse zeigen, dass das Modell in der Lage ist, unter allen Testbedingungen erfolgreich zu generalisieren und dabei ein systematisches Verhalten zeigt. Darüber hinaus weisen die verbleibenden Fehler starke Ähnlichkeit zu anderen semantischen Repräsentationen auf. Dies findet sich in der Literatur zu Sprachfehlern wider, wo es heißt, dass Fehler Elemente semantischer oder phonologischer Ähnlichkeit beinhalten. Dieses Ergebnis beweist das

systematische Verhalten des Modells, da es ähnliche semantische Repräsentationen in ähnlicher Weise verarbeitet, auch wenn sie dem Modell unbekannt sind.

Zweitens wurde das Satzproduktionsmodell auf zwei verschiedene Arten analysiert. (i) Indem man sich die von ihm erzeugten Sätze ansieht, einschließlich der aufgetretenen Fehler, und dabei die Schwierigkeiten und Präferenzen des Modells hervorhebt. Die Ergebnisse zeigen, dass das Modell die syntaktischen Muster der Sprache lernt. Darüber hinaus zeigt sich, dass die verbleibenden Probleme im Wesentlichen mit sehr ähnlichen semantischen Repräsentationen zusammenhängen, die manchmal ungewollte Sätze produzieren, welche jedoch semantisch nah an den beabsichtigten Sätzen liegen. (ii) Indem die Verbindungsgewichte und Aktivierungsmuster des Modells analysiert und eine algorithmische Darstellung der internen Verarbeitung erzielt wird. Demnach aktiviert die semantische Eingangsrepräsentation jene Wörter, mit denen sie inhaltlich zusammenhängt. In diesem Zusammenhang wird ein Ranking erzeugt, weil Wörter, die wahrscheinlich früh im Satz erscheinen eine stärkere Aktivierung erfahren. Im nächsten Schritt aktiviert das zuvor produzierte Wort syntaktische und semantische Einschränkungen der nächsten Wortproduktionen. Derweil speichern Kontext-Einheiten Informationen für einen längeren Zeitraum, und ermöglichen es dem Modell so, längere Abhängigkeiten zu realisieren. Nach unserem Verständnis können diese Erkenntnisse als Erklärungsgrundlage für andere, verwandte Modelle herangezogen werden.

Drittens wird eine Erweiterung des Modells vorgeschlagen, um die UID nachzubilden. Laut UID ist die Sprachproduktion ein effizienter Prozess, der von der Tendenz geprägt ist, linguistische Einheiten zu produzieren, die Informationen so einheitlich wie möglich verteilen, und dabei die Kapazität des Kommunikationskanals vor dem Hintergrund der sprachlichen Kodierungsmöglichkeiten ausreizt, wodurch die Menge der pro Zeiteinheit übertragenen Informationen maximiert wird. Dies wird in der Erweiterung umgesetzt, indem zwei verschiedene Strategien der Wortproduktion gegeneinander ausgespielt werden: Wähle das Wort (i) mit der höchsten Wahrscheinlichkeit unter den zuvor produzierten Wörtern; oder (ii) welches die Satzlänge minimiert. Durch die Kombination dieser beiden Strategien ist das Modell in der Lage, Sätze unter Vorgabe der Informationsdichte und -verteilung zu erzeugen, was einer ersten Modellierung der UID auf algorithmischer Ebene gleichkommt.

Zusammenfassend zeigen die Resultate, dass die verteilten semantischen Repräsentationen von Frank et al. (2009) für die Satzproduktion verwendet werden können und dabei Systematizität beobachtet werden kann. Darüber hinaus wird eine algorithmische Erklärung der internen Mechanismen des Modells geliefert. Schließlich wird ein Modell der UID vorgestellt, das einen ersten Schritt zu einer mechanistischen Darstellung auf der algorithmischen Ebene der Analyse darstellt.

Ausführliche Zussamenfassung

Als *menschliche Sprachproduktion* bezeichnet man den Prozess, bei dem eine Person ihre Gedanken M in eine sprachliche Äußerung U kodiert, so dass, wenn eine andere Person U diese hört oder liest, sie in der Lage wäre, eine mentale Repräsentation M' der Äußerung zu bilden. Diese Repräsentation sollte idealerweise so nah wie möglich an den ursprünglichen Gedanken M liegen. Die spezifischen Mechanismen, die in einem solchen Prozess involviert sind, wurden aus verschiedenen Perspektiven untersucht und haben zu mehreren Berichten und Modellen geführt, die versuchen dieses Phänomen zu erklären. Diese Dissertation widmet sich der Aufgabe, die menschliche Sprachproduktion aus einer konnektionistischen Perspektive zu modellieren, indem sie rekurrente künstliche neuronale Netze und verteilte semantische Repräsentationen verwendet.

Konkret befassen wir uns mit der Sprachproduktion auf Satzebene und konzentrieren uns auf den Prozess der Umwandlung einer zu vermittelnden Botschaft oder semantischen Repräsentation in eine Folge von Wörtern, die einen Satz bilden. Dies geschieht ohne Rücksicht auf die Prozesse, die während des Aufbaus der Botschaft beteiligt sein könnten, oder diejenigen Prozesse, die an der phonologischen Kodierung oder Artikulation beteiligt sind.

In diesem Zusammenhang werden in dieser Arbeit drei Hauptthemen behandelt:

a) die Verwendung von verteilten Repräsentationen zur konnektionistischen Modellierung der Sprachproduktion in Bezug auf Systematizität,

b) die interne Dynamik von Sprachproduktionsmodellen mit rekurrenten neuronalen Netzen,

c) die Implementierung eines Modells, das die Intuitionen der "Uniform Information Density Hypothesis" widerspiegelt (UID; Jaeger, 2006; Levy and Jaeger, 2007).

Hinsichtlich des ersten Punktes basieren die hier verwendeten Repräsentationen auf dem Distributed Situation Space Modell (DSS; Frank et al., 2003, 2009). Nach diesem Schema wird die Bedeutung eines Satzes in Bezug auf eine *Mikrowelt* dargestellt, die eine kleine

Gruppe von Entitäten enthält, welche mit probabilistischen Gesetzmäßigkeiten interagieren. Die resultierenden Repräsentationen entsprechen Punkten in einem Vektorraum, dessen Struktur durch die Regelmäßigkeiten der Mikrowelt gegeben ist. Folglich kann die Ähnlichkeit zwischen verschiedenen Repräsentationen beurteilt werden, indem man sich die Repräsentationen selbst ansieht, so dass ein Modell, das diese Bedeutungsrepräsentationen verwendet, eine Interpretation der Repräsentation vornehmen kann. Dies ist auch tannier Fall, wenn Repräsentationen neu für das Modell sind. Darüber hinaus enthalten DSS-Repräsentationen sehr reichhaltige probabilistische Informationen über die Situationen, in denen ein Satz wahr ist, und ermöglichen so eine Schlussfolgerung basierend auf Weltwissen. Zum Beispiel würde die Nachrichtenrepräsentation für den Satz "Charlie spielt Fußball"' Informationen über die Orte, an denen Fußball gespielt werden kann, mögliche Spieler, Gewinner, etc. enthalten.

DSS-Repräsentationen wurden ursprünglich entwickelt, um semantische *Systematizität* im Sprachverständnis (Frank et al., 2009) zu demonstrieren. Systematizität bezieht sich auf die Fähigkeit, von einer Reihe bekannter Instanzen zu neuen Instanzen zu verallgemeinern und von den Gemeinsamkeiten zwischen den bekannten und den neuen Instanzen zu profitieren. Es wurde vorgeschlagen, dass dies ein allgemeines Merkmal und sogar ein Gesetz der kognitiven Systeme (Fodor and McLaughlin, 1990; Fodor and Pylyshyn, 1988) sei. Ihre Bedeutung liegt darin, dass das Auswendiglernen aller Nachrichten-Satz-Paare nicht möglich ist, da sowohl die Anzahl der möglichen Nachrichten als auch die Anzahl der möglichen Sätze unendlich ist und daher ein Modell der Satzproduktion oder des Satzverständnisses zwangsläufig eine Verallgemeinerung aufweisen muss.

Im Vergleich zur Wortproduktion, bei der lexikalische Elemente meist aus dem Gedächtnis abgerufen werden (mit Ausnahme von Wörtern, die durch morphologische Produktivität erzeugt werden), ist die Satzproduktion zudem ein viel produktiverer Prozess. Während die Anzahl der lexikalischen Elemente in einer Sprache begrenzt ist und die meisten Wörter einem Sprecher bekannt sind, ist die Anzahl der Sätze, die mit derselben Sprache erzeugt werden können, unendlich. Basierend darauf sind viele Sätze für einen Sprecher neuartig. Daher ist die Systematizität von größter Bedeutung für ein Modell der Satzproduktion oder des Verständnisses.

Diese Dissertation testet, ob DSS-Repräsentationen auch zur Modellierung der Sprachproduktion verwendet werden können und demonstriert Systematizität wie Frank et al. (2009). Um dies zu erreichen, präsentieren wir ein Modell der Satzproduktion, das DSS-Vektor-Repräsentationen als Input verwendet und ansonsten eine sehr ähnliche Architektur wie das Prod-SRN-Modell von Chang (2002) hat.

Das vorgeschlagene Modell besteht aus einer Erweiterung eines "Simple Recurrent Neural Network" (SRN, Elman, 1990). Mit jeder Zeiteinheit erhält eine rekurrente Ebene als

Eingabe die DSS-Repräsentation des zu übertragenden Satzes und des Wortes, das im vorherigen Zeitpunkt erzeugt wurde (leer zum Zeitpunkt 0), dann wird die Aktivierung der rekurrenten Ebene einer Softmax-Ebene zugeführt, wobei jede Einheit in dieser Ebene einem Wort im Wortschatz entspricht. Zu jedem folgenden Zeitpunkt wird das Wort mit der höchsten Aktivierung in der Ausgangsebene erzeugt, das zum nächsten Zeitpunkt der rekurrenten Ebene zugeführt wird. Dieser Prozess wird fortgesetzt, bis das Modell einen Punkt erzeugt.

Dieses Modell wurde unter verschiedenen Testbedingungen in Bezug auf verschiedene Verallgemeinerungsebenen unter Verwendung der ursprünglichen DSS-Repräsentationen von Frank et al. (2009), auch *Situation Vektoren* genannt, evaluiert, während gleichzeitig eine alternative Repräsentationsform eingeführt wurde, die ebenfalls aus dem “Distributed Situation Space” abgeleitet ist, den wir *Belief Vektoren* nennen. Letztere wurden als alternative Methode eingeführt, um semantische Repräsentationen einer überschaubaren Dimensionalität zu erhalten, da während der Konstruktion der Situation Vektoren ein Dimensionalitätsreduktionsprozess einige Informationsverluste mit sich bringt. Diese führen dazu, dass einige Aspekte nicht mehr erkennbar sind. Die Ergebnisse zeigen, dass das Modell tatsächlich in der Lage war, Sätze mit beiden Arten von Nachrichtenrepräsentationen (Situation und Belief Vektoren) zu erzeugen, sich zu neuen Nachrichten zu verallgemeinern und eine sehr gute Leistung unter allen Bedingungen leistet. Darüber hinaus war das Modell in der Lage, nicht nur einen Satz zu produzieren, sondern auch die meisten der Sätze, die von der Grammatik erlaubt sind und die sich auf eine bestimmte Botschaft beziehen. Dies gilt auch auch wenn diese Botschaften neu für das Modell waren, was ein hohes Maß an Systematizität aufweist.

Eine Analyse der Ausgabe des Modells und der hervorgerufenen Fehler ergab, dass das Modell Schwierigkeiten mit Nachrichtenrepräsentationen hat, die sehr ähnlich sind und den Beweis der Sprachfehlerliteratur widerspiegeln, bei denen Fehler als Ergebnis semantischer oder phonologischer Ähnlichkeit auftreten. In diesem Fall sind die Fehler immer mit semantischer Ähnlichkeit verbunden, da das Modell nicht mit phonologischen Informationen arbeitet. Die Art solcher Fehler dient dazu, das systematische Verhalten des Modells weiter zu bestätigen, da ähnliche Nachrichtenrepräsentationen ähnlich verarbeitet werden, auch wenn sie neu für das Modell sind.

In Anbetracht der semantischen Repräsentationen, die als Input für das Modell verwendet wurden, besitzen diese einige Eigenschaften, die die Modellierung bestimmter Aspekte der menschlichen Sprachverarbeitung erleichtern, im Vergleich zu symbolischen diskreten Repräsentationen. Erstens ist jede Repräsentation ein multidimensionaler kontinuierlicher Vektor, der einem Punkt in einem Vektorraum entspricht. Folglich ist die Anzahl der Repräsentationen, die aus diesem Raum gewonnen werden können, möglicherweise unendlich. Dies ist ein Attribut, das notwendig ist, wenn man unendliche Mengen darstellen will, wie beispielsweise die Menge der Sätze, die sich auf eine Sprache beziehen. Zweitens, diese Repräsentationen enthalten vollständige Beschreibungen der Situationen in der Mikrowelt, die sich auf jeden Satz beziehen. Dies ermöglicht direkte Schlussfolgerung und die Schätzung von probabilistischen Informationen im Zusammenhang mit den Ereignissen, die in jeder semantischen Repräsentation vermittelt werden. Schließlich ist eine weitere wichtige Eigenschaft von DSS-Repräsentationen, dass, da es sich um kontinuierliche Vektoren handelt, die Ähnlichkeit zwischen den Repräsentationen einfach durch Messen des Abstands zwischen den zugehörigen Vektoren bewertet werden kann. Diese Eigenschaft ist wichtig, da die Eingangsrepräsentationen untereinander vergleichbar sein müssen, damit ein Modell Beziehungen von bekannten zu neuen Inputs herstellt und so eine Verallgemeinerung erreicht.

Aufgrund dieser Eigenschaften können Situation und Belief Vektoren als gute Kandidaten für die Modellierung der menschlichen Sprachverarbeitung angesehen werden, in diesem Fall für die Sprachproduktion im Vergleich zu symbolischen diskreten Repräsentationen. Dies spiegelt sich in den positiven Ergebnissen der Simulationen widers.

Während heutzutage rekurrente neuronale Netze und Deep-Learning-Architekturen im Allgemeinen erfolgreich für eine Vielzahl von Aufgaben eingesetzt werden (siehe z.B. LeCun et al., 2015), ist ihr interner Mechanismus nicht vollständig verstanden und wird eher als Blackbox verwendet. In unserem Fall zeigte das Produktionsmodell, dass es in der Lage war, korrekte Sätze mit Systematizität zu produzieren, wobei der interne Produktionsmechanismus des Modells noch ziemlich unklar war. Aus diesem Grund wurde eine Analyse der internen Dynamik des Modells durchgeführt.

Die Analyse basierte auf Layer-wise Relevance Propagation (Bach et al., 2015). Dieser Algorithmus ähnelt dem Backpropagation-Algorithmus (Rumelhart et al., 1986), er beginnt bei der Output-Ebene und bewegt sich im Diagramm in Richtung der Input-Einheiten, wobei er die Relevanz verfolgt, die jede Einheit in der Ebene l_{i-1} bei der Aktivierung von Einheiten in der Ebene l_i hat, zurück zu den Input-Einheiten, die normalerweise von Menschen interpretierbar sind.

Die Ergebnisse zeigen, dass jedes Input-Muster die Aktivierung von Wörtern fördert, die mit seiner Semantik zusammenhängen, während es die Wörter hemmt, die sich im Konflikt befinden. Nach der Produktion jedes Wortes werden syntaktische und semantische Beschränkungen eingeführt, die die folgenden Wortproduktionen beeinflussen, während die Kontext-Einheiten Informationen im Laufe der Zeit bewahren. In dieser Ansicht kann das Verhalten des Modells als eine Reihe von Einheiten erklärt werden, bei denen jede Einheit eine bestimmte Funktion hat und das Modell die richtigen Wechselwirkungen zwischen den Einheiten lernt, so dass ein korrektes globales Verhalten entsteht. Es ergibt sich dann eine korrekte Verarbeitung neuartiger Inputs, wenn die während des Trainings erlernten Regeln, die die Wechselwirkungen zwischen den Einheiten regeln, mit den Regeln der Produktionsaufgabe übereinstimmen.

Die Analyse wies auch auf Muster hin, die zuvor in Bezug auf die Sprachverarbeitung mit rekurrenten neuronalen Netzen vorgeschlagen wurden: Einige versteckte Einheiten schienen mit spezifischen Funktionen (Karpathy et al., 2015) zusammenzuhängen. Die Aktivierungsmuster, die mit jedem Wort zusammenhängen, spiegeln die Ähnlichkeit zwischen den Wörtern (Mikolov et al., 2013) wider und darüberhinaus die allgemeine Intuition, dass das Wiederauftreten dazu dient, Informationen über Zeit zu erhalten. Die hier vorgestellten Ergebnisse demonstrieren solche Intuitionen und liefern eine mechanistisch ganzheitliche Darstellung der Funktionsweise des Modells.

Aufgrund von architektonischen Ähnlichkeiten mit anderen Sprachmodellen erwarten wir, dass dieser Mechanismus das Verhalten ähnlicher Modelle der Sprachproduktion widerspiegelt (z.B. Chang, 2002; Chang et al., 1997; Dell et al., 1993), sowie größere Modelle, die in der Computerlinguistik verwendet werden, wie sie beispielsweise in der Sprachmodellierung (z.B. Mikolov et al., 2010) oder der maschinellen Übersetzung (z.B. Sutskever et al., 2014) verwendet werden. Die hier beschriebene Methodik könnte auch dazu dienen, eine solche Hypothese in zukünftigen Arbeiten zu testen.

Im Hinblick auf die Implementierung eines Modells der UID wird eine Erweiterung des Modells mit dem Ziel vorgestellt, eine mechanistische Darstellung der Uniform Information Density Hypothesis (UID; Jaeger, 2006, 2010; Levy and Jaeger, 2007) zu erhalten. UID stellt fest, dass die Sprachproduktion ein effizienter Prozess ist, der von der Tendenz geprägt ist, linguistische Einheiten zu produzieren, die die Informationen so einheitlich wie möglich und nahe an der Kapazität des Kommunikationskanals verteilen, unter Berücksichtigung der Kodierungsmöglichkeiten der Sprache, wodurch die Menge der Informationen, die pro Zeiteinheit übertragen wird, optimiert wird. Beweise für eine solche Tendenz wurden zwar vorgelegt (z.B. Bell et al., 2003; Jaeger, 2006), jedoch wurde kein Ansatz vorgeschlagen, der erklären würde wie ein solcher Mechanismus algorithmisch durch ein Produktionsmodell implementiert würde.

Diese Erweiterung war in der Lage, die beiden wichtigsten rationalen Ziele zu implementieren, die zur Steuerung von UID vorgeschlagen wurden: Auf der einen Seite kann das Modell eine Tendenz zeigen, die Menge der pro Zeiteinheit übertragenen Informationen zu maximieren, wobei es schnell ist; und auf der anderen Seite versucht das Modell, die Grenze der Kanalkapazität zu unterschreiten, um Misskommunikation zu vermeiden. Durch das Ausbalancieren dieser beiden Tendenzen könnte man verschiedene Ebenen der Einheitlichkeit erreichen, die einen Ausgangspunkt für das Modellieren von UID auf der algorithmischen Ebene der Analyse bilden.

Während der Entwicklung des UID-Modells traten einige Probleme auf, die einige Aspekte hervorhoben, die weiter untersucht werden müssen. Erstens unterscheidet sich die Informationsverarbeitung aus Sicht des Empfängers von der Sicht des Produzenten: Der Produzent kennt vorher die Botschaft, die er zu vermitteln versucht, während der Empfänger die beabsichtigte Botschaft aus dem erzeugten Satz ableiten muss. Daraus folgend bezieht sich der Produktionsaufwand mehr auf den Zugang der linguistischen Einheiten, die zum Aufbau des Satzes benötigt werden, während der Aufwand des Verstehens mehr auf das Erkennen der Einheiten und den Aufbau einer mit dem Satz kohärenten Nachrichtendarstellung bezogen wäre. Zweitens, wenn man bedenkt, dass der Informationsgehalt, der sich auf einen bestimmten Satz bezieht, sowohl mit der Wahrscheinlichkeit des Ereignisses, das der Satz vermittelt, als auch mit der Wahrscheinlichkeit der linguistischen Elemente, die zur Bildung des Satzes verwendet werden zusammenhängt, ist es notwendig einen Datensatz zu erstellen, der die relative Verteilung der Ereignisse und ihrer jeweiligen Sätze auf natürliche Weise widerspiegelt, so dass ein auf diesem Datensatz trainiertes Modell in der Lage ist abzuleiten, wie wahrscheinlich ein bestimmtes Ereignis ist und wie wahrscheinlich eine bestimmte Folge von linguistischen Einheiten ist. Schließlich ist es notwendig, die Obergrenze oder Kapazität des Kommunikationskanals zu definieren. Ein Satz kann ein sehr einheitliches Überraschungsprofil haben, während er sehr wenig oder sehr viele Informationen pro Zeiteinheit übermittelt. Einheitliche Überraschungsprofile sind zwar wünschenswert, aber sie müssen in Bezug auf die Kanalkapazität einheitlich sein, was bis heute nicht klar ist.

Im Allgemeinen zeigen die Ergebnisse dieser Arbeit, dass in der Tat die verteilten semantischen Repräsentationen von Frank et al. (2009) zur Modellierung der Satzproduktion verwendet werden können, wobei sie Systematizität und im Allgemeinen eine hohe Performance in allen Simulationen aufweisen. Darüber hinaus zeigt das vorgestellte Modell ein Verhalten, das Tendenzen widerspiegelt, die von der Sprachfehlerliteratur berichtet wurden, sowie Hinweise auf den Einfluss statistischer Muster auf die Sprachproduktion. Da das Modell eine relativ geringe Dimensionalität hat und die Struktur des Datensatzes bekannt ist, könnte man sein internes Verhalten analysieren, um Einblicke in die allgemeine Dynamik rekurrenter neuronaler Netze zu erhalten, die zu einer algorithmischen Darstellung des Verhaltens des Modells führen. Schließlich, profitierend von der Fähigkeit des Modells mehrere Kodierungen für eine gegebene Semantik zu produzieren, wird ein Modell der UID vorgestellt, das in der Lage ist, verschiedene Produktionsstrategien zu implementieren, und das einige wichtige Aspekte der UID hervorhob. Dazu gehört die Notwendigkeit, die Kapazität der Kommunikationskanäle und die Unterschiede in der Verarbeitung zwischen Produktion und Verständnis zu definieren, um von der Formulierung der UID auf der Berechnungsebene der Analyse zu einer mechanistischen Darstellung auf der algorithmischen Ebene zu gelangen.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Humans are inherently social beings who need communication in order to coordinate, live together and ultimately form complex societies. This communication is mostly achieved through language, an ability that has been regarded as a characteristic that differentiates humans from other animals. Indeed, we use language everyday to communicate almost any kind of information. In spite of its importance, some aspects of this phenomenon still need further study, amongst them, the cognitive aspects of language use: how people produce and understand language.

In order to study language from a cognitive perspective, we can divide it in two major processes: *language production*, where a person encodes his/her thoughts M into a linguistic utterance U; and *language comprehension*, where a person decodes a linguistic utterance U into a mental representation M' of the information the speaker intended to communicate. In this process, the utterance produced by the speaker U shoould be such that the message that the comprehender is able to decode M' is as similar as possible to the message the speaker intended to communicate M. In this view, the speaker is responsible for efficiently providing enough information under the environmental constraints, in order for the comprehender to be able to decode the intended message.

Concerning human language production, a body of evidence has been gathered trying to describe and analyze this process. The methods that have been utilized include the inspection of speech errors that people elicit during normal conversation (e.g., Meringer and Mayer, 1895), the recording of reaction times and behavioral preferences during a production-related task (e.g., Fraisse, 1967), and more recently the recording of neural activity also during a production-related task (e.g., Graves et al., 2007). This evidence has served to motivate a number of accounts and computational models that try to explain the process of human language production.

In general, the models that have been proposed assume a sequence of steps, which are typically linked to different levels of linguistic representation. These levels were originally motivated by the regularities of speech errors elicited during conversation, as errors tend to involve elements of the same linguistic level. For example, word exchanges can span some distance and tend to occur only between complete words of the same syntactic category, while sound exchanges ignore syntactic category and involve words that are close to each other (Garrett, 1975). Additionally, word substitution errors usually involve words that are semantically similar, suggesting that at an abstract "semantic level", similar words can be confused, possibly provoking the production of an unintended but semantically similar word (Fromkin, 1973).

Amongst these models, the one proposed by Levelt (1989) has been one of the most influential, serving as reference for language production research. It describes the complete process from intention to articulation of speech, based on the empirical evidence available at the time of publication. According to Levelt (1989), language production can be segmented in three modules:

- A conceptualizer, which generates the message to be conveyed, and monitors what has been and what is about to be said.
- A formulator, which during *grammatical encoding* translates the message into a sequence of lemmas following the syntactic rules of the language; and then, during *phonological encoding*, this sequence of lemmas is translated into a phonetic/articulatory plan.
- An articulator, which specializes in the motor execution of the phonetic plan, producing overt speech.

This dissertation focuses on language production at a sentence level, and more specifically, on the process of converting a message to be conveyed, or semantic representation, into a sequence of words forming a sentence; without considering the processes that might be involved during the construction of the message, or the processes involved in phonological encoding or articulation.

In the model of Levelt (1989), this corresponds to grammatical encoding. During grammatical encoding, according to Levelt (1989), the lemmas in the mental lexicon whose meaning match part of the message are activated, making their syntax available and thus activating syntactic building procedures. Using the latter, syntactic structures are built, such as verb or noun phrases. After the relevant lemmas have been retrieved and the relevant syntactic procedures have finished, the grammatical encoder would have produced a *surface structure*, an ordered string of lemmas that follows the syntactic rules of the language.

Compared to word production where lexical items are mostly retrieved from memory (with the exception of morphological productivity), sentence production is a much more productive process. While the number of lexical items in a language is finite and most words are known by a speaker, the number of sentences that can be generated with the same language is infinite, and therefore many sentences are novel to a speaker. As a consequence, an important attribute that a model of sentence production or comprehension has to exhibit is *systematicity*, which is the ability to generalize from a set of known instances to novel ones, profitting from the commonalities between the known and the novel instances. Its importance lies on the fact that memorization of all message-sentence mappings is not possible, as both the number of possible messages and the number of possible sentences are infinite, and consequently, a model of sentence production or comprehension necessarily has to show generalization. Furthermore, systematicity has been proposed to be a general characteristic and even a law of cognitive systems (Fodor and McLaughlin, 1990; Fodor and Pylyshyn, 1988).

The degree to which a model can generalize is related to the complexity of the task. During sentence production not only the correct words have to be retrieved from memory, but also they have to be in the correct order, meeting the syntactic constraints of the language while conveying the information of the message. Therefore, a model of sentence production has to learn, on the one hand, the way lexical items can be combined according to the syntactic rules of the language; and on the other hand, with this syntactic knowledge, how to translate a message onto a sequence of lexical items.

Moreover, a model of human language production is valid to the extent that it exhibits a behavior and internal processing similar to humans. Then, it is not only necessary that a model of sentence production learns to produce sentences in a systematic way, but also, the model has to approximate the mechanism that humans utilize to solve the same task, reflecting the evidence regarding human language production.

Considering these aspects, some connectionist models of sentence production have been postulated. Connectionist models are inspired by the way the brain works. They follow the principle that mental phenomena can be described by interconnected networks of simple units, i.e., artificial neural networks. These models have been criticized with claims that they cannot exhibit systematicity to the level that humans do (Fodor and Pylyshyn, 1988). Consequently and given its importance for language production, some connectionist models of language production have focused on demonstrating their generalization capacity. Most prominent amongst them are the Structural Priming model (Chang et al., 1997), the Prod-SRN model (Chang, 2002) and the Dual-Path model

(Chang, 2002), which implement an extension of a Simple Recurrent Neural network (Elman, 1990; Jordan, 1986), being able to produce sentences with different degrees of generalization.

The Structural Priming model and the Prod-SRN model have a very similar architecture in which a message representation is given as input to a recurrent layer that in turn activates an output layer containing lexical items. In contrast, the Dual-Path model presents a much more complex architecture that was partially motivated by the low level of systematicity that the Prod-SRN model showed during the experiments conducted by Chang (2002). Its architecture is formed by two different paths of computation that converge on a recurrent layer, and such that the recurrent layer can interact only with semantic roles, that in turn are connected to concepts of the message representation. This separation between semantic roles and concepts allows the Dual-Path model to exhibit higher levels of systematicity compared to the Prod-SRN model (Chang, 2002).

An aspect that these 3 models share is the use of localist input units, where the message to be conveyed is represented by a set of binary units where only few of them are activated signaling specific symbolic features. In contrast, this dissertation explores the use of distributed message representations, where a message is represented by a point in a multidimensional continuous meaning space. Using these, we test whether one can achieve high levels of systematicity without increasing the complexity of the architecture.

The representations used here are based upon the Distributed Situation Space model (DSS; Frank et al., 2003, 2009). Under this scheme, the meaning of a sentence is represented with respect to a micro-world, which is a small set of entities that interact with probabilistic regularities. The resulting representations correspond to points in a vector space whose structure is given by the regularities of the micro-world. In consequence, similarity can be assessed between different representations by looking at the representations themselves, permitting a model using these meaning representations to interpret a representation even if it is new to the model. Moreover, DSS representations contain very rich probabilistic information about the situations in which a sentence is true, allowing for 'world-knowledge'-driven inference. For example, the message representation for the sentence "Charlie plays soccer." would contain information about the locations where soccer can be played, possible players, winners, losers, etc.

DSS representations were originally developed to demonstrate semantic systematicity in language comprehension (Frank et al., 2009), and have been successfully used to model language comprehension (e.g., Frank et al., 2009; Venhuizen et al., 2018). In this dissertation, we test whether these representations can also be used to model language production, demonstrating systematicity as in Frank et al. (2009). In order to achieve that, we present a model of sentence production that uses DSS representations as input, and that otherwise has a very similar architecture to the Prod-SRN model of Chang (2002).

The proposed model of sentence production was evaluated under different test conditions related to different levels of generalization. The results show that indeed the model was able to produce sentences using the DSS message representations, generalizing to new messages, exhibiting a very good performance in all conditions. Furthermore, the model was able to produce not only one sentence, but most of the sentences that are allowed by the grammar and that are related to a given message, even if the latter was novel to the model, demonstrating a high level of systematicity.

An analysis of the output of the model and the mistakes elicited revealed that the model has difficulties with message representations that are highly similar, reflecting the evidence of the speech error literature, where errors arise as a result of semantic or phonological similarity. In this case, the errors are always related to semantic similarity, since the model does not operate with phonological information. Moreover, the nature of such mistakes serves to corroborate the systematic behavior of the model, as similar message representations are processed similarly, even if they are new.

While nowadays recurrent neural networks, and deep learning architectures in general, are used successfully for a large variety of tasks (see e.g., LeCun et al., 2015), their internal mechanism is not fully understood and are rather used as a black box. It is evident that it is not sufficient to present a cognitive model that imitates the experimental evidence, but also, one has to understand its internal mechanism. In order to look into the internal mechanism of this model in particular, an analysis of the connection weights and activation patterns was performed. Such an analysis permitted us to obtain an algorithmic account of the behavior of the model that could be generalized to other language production models with similar architecture (e.g., Chang, 2002; Chang et al., 1997; Dell et al., 1993), as well as those used in computational linguistics for language modeling (e.g., Mikolov et al., 2010), caption generation (e.g., Chen and Lawrence Zitnick, 2015), or machine translation (e.g., Sutskever et al., 2014).

Finally, since the model can produce several encodings for a given message representation, one can manipulate its configuration in order to modify its production preferences. In that way, an extension of the model is presented with the goal of obtaining a mechanistic account of the Uniform Information Density Hypothesis (UID; Jaeger, 2006, 2010; Levy and Jaeger, 2007). UID states that language production is an efficient process affected by a tendency to produce linguistic units distributing the information as uniformly as possible and close to the capacity of the communication channel, given the encoding possibilities of the language, thus optimizing the amount of information that is transmitted per time unit. Evidence for such a tendency has been presented (e.g., Bell et al., 2003; Jaeger, 2006), however, no account has been proposed that would explain how such a mechanism would be algorithmically implemented by a production model. While the UID model presented here needs further development, it serves to highlight some important concerns that are still open regarding UID.

In general, the results of this dissertation, which are elaborated in the next chapters, show that the distributed representations of Frank et al. (2009) can be used to model language production exhibiting systematicity (see Chapter 5). Moreover, the model presented shows a behavior reflecting tendencies reported by the speech error literature, as well as evidence suggesting the influence of statistical patterns on language production. Since the model has a relatively low dimensionality and the structure of the dataset is known, one can analyze its internal behavior giving insights about the general dynamics of recurrent neural networks (see Chapter 6). Finally, profiting from the ability of the model to produce several encodings for a given semantics, a model of UID is presented that is able to implement different production strategies and that highlighted some important aspects about UID, such as the need to define the communication channel capacity, and the differences in processing between production and comprehension (see Chapter 7).

1.1 Contributions

More concretely, the main contributions that this dissertation proposes are the following:

- An alternative scheme for semantic representations, called *belief vectors*, that is derived from the Distributed Situation model (Frank et al., 2003, 2009).
- A model of sentence production that uses as input the aforementioned representations, as well as the situation vectors defined by Frank et al. (2009).
- An evaluation and analysis of the behavior of the sentence production model, showing that it was able to learn the language and furthermore it was able generalize to novel sentences and semantic representations, exhibiting syntactic and semantic systematicity.
- An analysis of the internal mechanism of the sentence production model by inspecting its connection weights and average activations, reaching an algorithmic account of the behavior of the model that could possibly generalize to other models with similar architectures.

- A mechanistic account of the Uniform Information Density Hypothesis (UID; Jaeger, 2006; Levy and Jaeger, 2007), by extending the sentence production model previously presented.

1.2 Organization

This dissertation is organized in 8 chapters, including the introduction. The second chapter contains the Background, describing some of the most influential models of language production and their motivations. It also introduces the main ideas of the Uniform Information Density Hypothesis (UID; Jaeger, 2006; Levy and Jaeger, 2007). The third chapter, Semantic Representations, describes the type of semantic representations that are used as input for the model of sentence production. The fourth chapter, Sentence Production Model, describes the architecture and the training procedure of the model of sentence production. The fifth chapter, Semantic Systematicity, presents experiments using the model previously described, testing its capability regarding generalization or systematicity. The sixth chapter, Sentence Production Dynamics, presents an analysis of the internal mechanism of the model, by inspecting its connection weights and activation patterns. The seventh chapter, Approximating UID, presents an extension of the model that represents a first attempt to model UID at an algorithmic level of analysis. Finally, the eight chapter is devoted to conclusions and future work.

Chapter 2

Background

In this chapter we briefly overview the literature about accounts of Human Language Production, emphasizing those that are most relevant for this work. Afterwards, we also overview the Uniform Information Density Hypothesis (UID, Jaeger, 2006, 2010; Levy and Jaeger, 2007) which we will address in the last part of the dissertation. These elements will serve to set up a context in order to better understand and motivate the contributions made in the following chapters.

2.1 Accounts of Human Language Production

We take *Language Production* to refer to the process performed by humans when encoding an idea or message representation into an utterance conveying such information, in contrast to *Language Comprehension*, where an utterance is mapped to a message representation. These two skills permit humans to communicate in a seemingly effortless way.

Language production is a very rapid process: in a normal fluent conversation we produce 2-3 words per second, corresponding to about 4 syllables and 10 or 12 phonemes (Levelt, 1999). Each word is selected from a large vocabulary, typically containing 50-100 thousand words in a normal literate person (Miller, 1991). In spite of this speed, errors are rare: one is produced no more than once or twice in 1000 words (Garnham et al., 1981). Language production is also one of the most exercised human skills: in less than 40 minutes of talking a day, we will have produced around 50 million words by the time we reach adulthood (Levelt, 1999).

The study of this human phenomenon has been approached from a variety of perspectives, including the analysis of speech errors, reaction times, production preferences, and neurophysiological activity. The results of these approaches have led to the proposal of several models and accounts of human language production, each one seeking to explain some aspects of the available empirical evidence. In this section we will briefly review some of the most influential models and accounts, while introducing the empirical evidence that inspired them.

2.1.1 Speech Errors

Some of the earliest investigations of language production analyzed the errors that people exhibit normally in conversation, assuming that the nature of these errors would give insight into the mechanism that generated them.

Meringer and Mayer (1895) presented a large collection of German speech errors, which permitted the distinction between errors related to form ("Studien" instead of "Stunden") and to meaning (e.g., "ihre" instead of "meine"), with a large proportion related to both. An important aspect that was later recognized is that these errors very often involve some notion of *similarity*. That is, errors related to form often concern elements with similar form, and likewise, errors related to meaning concern elements with similar meanings.

Specific types of errors include (Carroll, 2007):

- *exchanges* (e.g. "mell wade" for "well made")
- *anticipations* (e.g. "taddle tennis" for "paddle tennis")
- *perserverations* (e.g., "been abay" for "been away")
- *blends* or *contaminations* (e.g., "evoid", blending "avoid" and "evade")
- *additions* (e.g., "moptimal" for "optimal")
- *deletions* (e.g., "pecific" for "specific")
- *shifts* (e.g., "she decide to hits it" for "she decides to hit it")
- *substitutions* (e.g."tennis bat" for "tennis racquet")

Given the regularities of the reported errors, as they generally involve elements with some relation to the intended utterance, Meringer and Mayer (1895) proposed three reasons for them:

1. interference from intended elements of the utterance

2. interference from an alternative formulation of the intended thought
3. interference from an unintended thought

where interference may be caused by competition at different levels of representation (see Butterworth, 1981).

Some errors related to the third reason correspond to what Freud (1924) described and linked to interfering unconscious thoughts; however, these errors also include cases where the speaker is well aware of alternative competing ideas, or where there are external stimuli interfering with the intended message (e.g., Garrett, 1980; Hill, 1973; Meringer and Mayer, 1895). Some authors have tried to integrate these errors into their own theoretical frameworks (e.g., Butterworth, 1980), some others have tried to reduced them to instances of the first two reasons (e.g., Ellis, 1980; Timpanaro, 1975); but in general, research has been focused on the first two reasons as the corresponding errors seem to be the most common and less controversial.

Another possible source of errors that has been postulated, in contrast to interference, corresponds to cases where noisy communication between different processing modules introduces addressing errors. For example, for the case of sound related substitutions, Fay and Cutler (1977) proposed that phonological items are organized in a phonological basis (e.g. all one-syllabled words beginning with /t/ are grouped together in similar areas, all three-syllabled words beginning with /k/ are also grouped together and so on.). Then, during retrieval, slight changes or mutations on the address of the intended element would result on the retrieval of distinct but similar elements to the intended ones.

2.1.2 Utterance Generator (Fromkin, 1971)

The first attempt to relate speech errors in a systematic way to an integrated linguistic theory was made by Fromkin (1971). She took the speech error findings available at the time and related them to generative grammar. The result was a theoretical speech production model, called Utterance Generator, whose features ranged from semantics to phonetics (see Figure 2.1).

In the first stage of this model, the message to be conveyed is generated. Then a syntactic structure is created, linking structural syntactic elements with semantic features. This structure is created prior to lexical selection in order to account for the fact that word switches occur only within and not across clauses (Fromkin, 1973). Additionally, creating a syntactic structure before lexical selection would account for the fact that

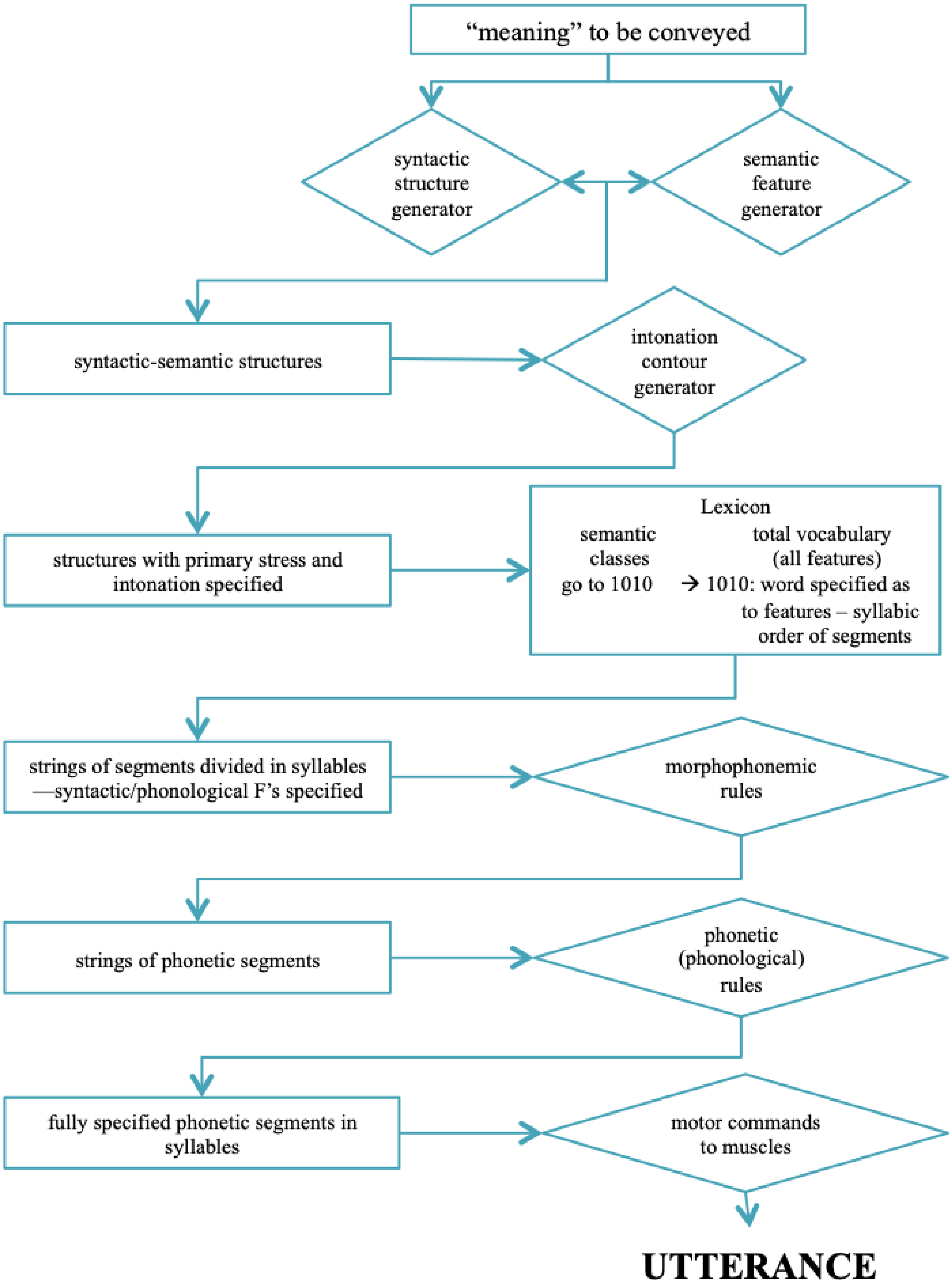
"meaning" to be conveyed
syntactic structure generator
semantic feature generator
syntactic-semantic structures
intonation contour generator
structures with primary stress and intonation specified
Lexicon
semantic classes go to 1010
total vocabulary (all features)
→ 1010: word specified as to features – syllabic order of segments
strings of segments divided in syllables —syntactic/phonological F's specified
morphophonemic rules
strings of phonetic segments
phonetic (phonological) rules
fully specified phonetic segments in syllables
motor commands to muscles
UTTERANCE

FIGURE 2.1: Utterance Generator. Adapted from Fromkin (1971).

word exchanges only occur between words of the same syntactic category. In the third stage, an intonation contour for the whole structure is generated. Having the intonation contour before lexical selection is intended to explain why the former does not seem to be affected by word exchanges. At the fourth stage, lexical selection occurs, starting with content words and then function words. This order is intended to explain the phonological accommodation that function words undergo with respect to the linguistic environment given by content words.

As one can see, this model is strictly top-down, where only one clause can be processed at a time, and where each stage has no access to higher or lower levels. While this model can account for a wide variety of speech errors, its strictly top-down nature fails to explain the *lexical bias*, which is the tendency of speech errors to form real words more often than non-words; among other phenomena that suggest more free interactions between stages.

Fromkin (1971) noticed that substitutions seem to occur when there is similarity either in form or in meaning between the intended element and an alternative one. This required the postulation of abstract features which do not show up in the final utterance but that are shared amongst the exchanged elements. Thus, she proposed abstract semantic features to explain meaning related substitutions, and thus, introduced the idea of the need for different levels of production or representation.

2.1.3 Model of Garrett (1975)

Garrett (1975) made more explicit this idea of levels of production or representation, noting that although speech errors can be elicited at all levels of linguistic representation, from phonemes to complete phrases, the elements involved tend to belong to only one level. For example, word exchanges can span some distance and mostly preserve the grammatical category and function within their clauses. Similarly, exchanges of sound or form (e.g. “rack pat” for “pack rat”) ignore grammatical category and occur between words that are close to each other. This suggested at least two different modular levels of production: one related to syntactic categories and another to the position of forms (morphemes, phonemes), which integrated the main components of his model of sentence production(Garrett, 1975, see Figure 2.2).

In this model, three levels are proposed:

1. Message Level: where the intended message is generated.
2. Sentence Level: where lexico-syntactic aspects of the sentence are defined.

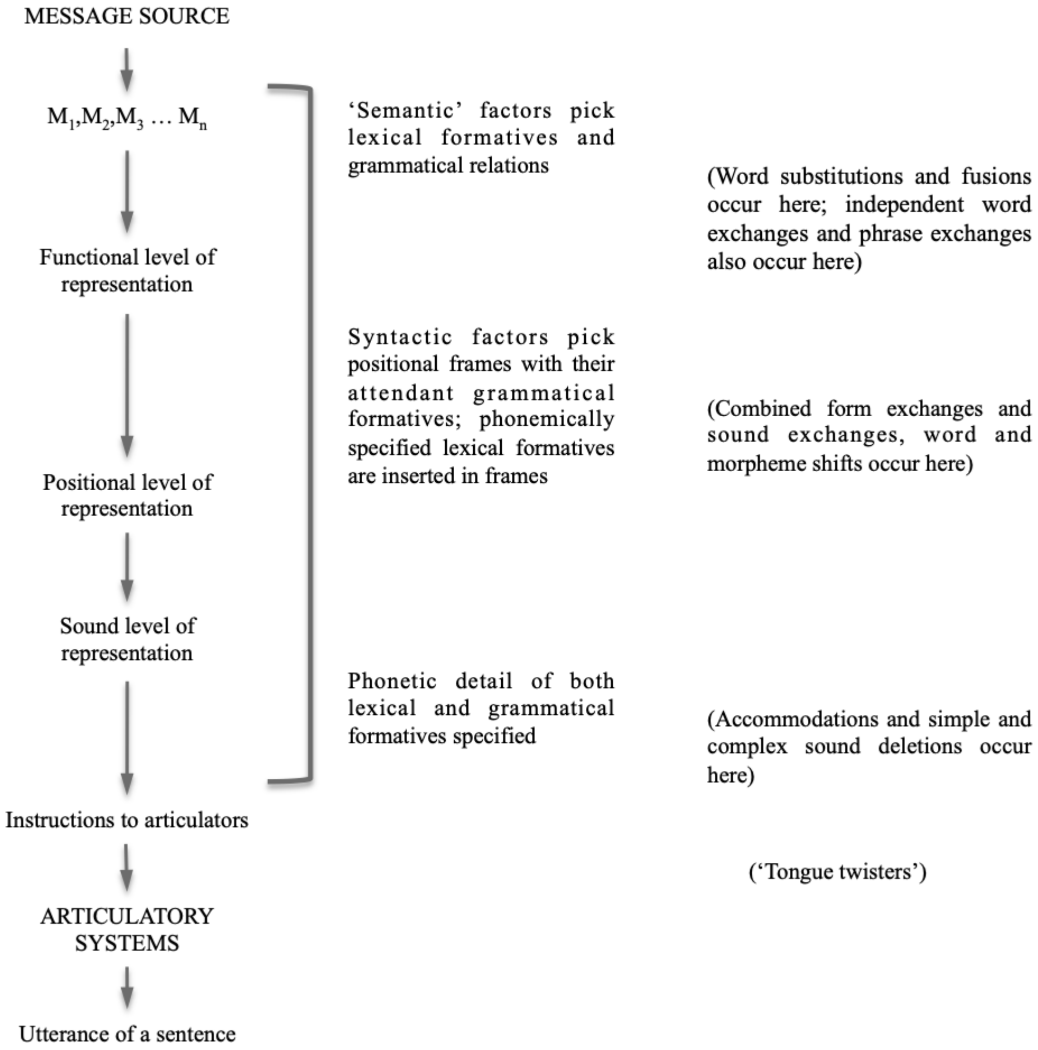
MESSAGE SOURCE
M1,M2,M3 … Mn
Functional level of representation
Positional level of representation
Sound level of representation
Instructions to articulators
ARTICULATORY SYSTEMS
Utterance of a sentence
'Semantic' factors pick lexical formatives and grammatical relations
Syntactic factors pick positional frames with their attendant grammatical formatives; phonemically specified lexical formatives are inserted in frames
Phonetic detail of both lexical and grammatical formatives specified
(Word substitutions and fusions occur here; independent word exchanges and phrase exchanges also occur here)
(Combined form exchanges and sound exchanges, word and morpheme shifts occur here)
(Accommodations and simple and complex sound deletions occur here)
('Tongue twisters')

Figure 2.2: Garret's model of sentence production. Adapted from Garrett (1975).

3. Articulatory Level: where instructions to the articulatory system are defined.

The Sentence Level is further divided into Functional and Positional levels, where the first one is related to lexical selection, as well as the syntactic constraints governing the selected words; and the second one is related to segment interactions and phonemic information. The separation of these two levels is motivated by speech error data. Meaning-related errors (e.g., word substitutions where an intended word is replaced by an unintended but semantically similar word with the same syntactic category) would occur at the Functional Level, while form-related errors (e.g. morpheme shifts or sound exchanges) would occur at the Positional Level. Additionally, this separation would explain phonological accommodation. Finally, and like Fromkin (1971), Garrett (1975) also emphasizes that content words are retrieved prior to function words, in view of some word exchanges where function words seem to adapt to content words even if the latter are exchanged.

Garrett's model of sentence production is very similar to that proposed by Fromkin (1971). In both cases, different levels of representation or processing are postulated, and the sequence of processing is strictly top-down, where each level dominates the immediate level down. Additionally, while Garrett (1975) does not commit to a single representation being processed at a time, allowing possibly for parallel processing, it is not clear how multiple parallel representations might interact with each other. Consequently, these models have difficulties explaining lexical bias, blend errors, errors concerning competition between alternative formulations of the intended thought, and errors where an unintended thought interferes with the intended one.

2.1.4 Spreading Activation Theory (Dell, 1986)

Similar to previous accounts, Dell (1986) also proposes different levels of processing. However, in this case the levels can interact not only top-down but also bottom-up. Moreover, each level is formed by a number of connected nodes representing distinct linguistic units (e.g., concepts, words, morphemes, phonemes, etc.) and following connectionist principles. This results in an account described as globally modular and locally interactive, where each two subsequent levels present interactions top-down and bottom-up (see Figure 2.3). Like the previous models, this one also focuses on explaining the evidence available regarding speech errors.

Although the theory contemplates a semantic level that specifies the meaning representation of the intended utterance, this theory describes mainly 3 levels of processing:

1. Syntactic: words are chosen and arranged according to the grammar rules.
2. Morphological: words are specified in terms of their constituent morphemes.
3. Phonological: words are spelled out in terms of their sound.

In each level there is a set of rules defining the combinatorial possibilities of units at that particular level; for example, at the syntactic level, the rules specify the syntactic categories and their combination within a sentence. Using these rules, a representation is constructed simultaneously at each level, with the rate of processing depending on each level and on the level immediately above it. In general, the selection of items for a lower representation must await the construction of the structures of the higher representation. Nonetheless, the degree to which processing at a higher level is ahead of the processing at a lower level can vary.

When a node is activated, activation spreads to all the nodes connected to it. Then the items with highest activation that follow the combinatorial rules are selected. After selection, their activation level immediately reduces to zero in order to avoid repetitions. According to this theory, speech errors are elicited because an incorrect item will sometimes have a higher activation than the correct one.

Given that the different levels interact flexibly with each other, a speech error can be the result of influence coming from multiple levels. As an example, Dell (1986) quotes someone saying “Let’s stop” when “Let’s start” was intended. In this case, one can recognize the semantic relation between the intended word and its substitution, however, one can also recognize phonological similarity, as the substitute word shares a common sound with the appropriate word. Errors of this nature have been shown to be overrepresented in speech errors corpora, where substitutions errors involve simultaneously meaning and phonological similarity (Dell and O’seaghdha, 1991; Harley, 1984).

2.1.5 Model of Levelt (1989)

The model presented in Levelt (1989) has become a point of reference in language production research. Considering the available evidence, the author gave a very detailed description of the process of speech production, from intention to articulation. The

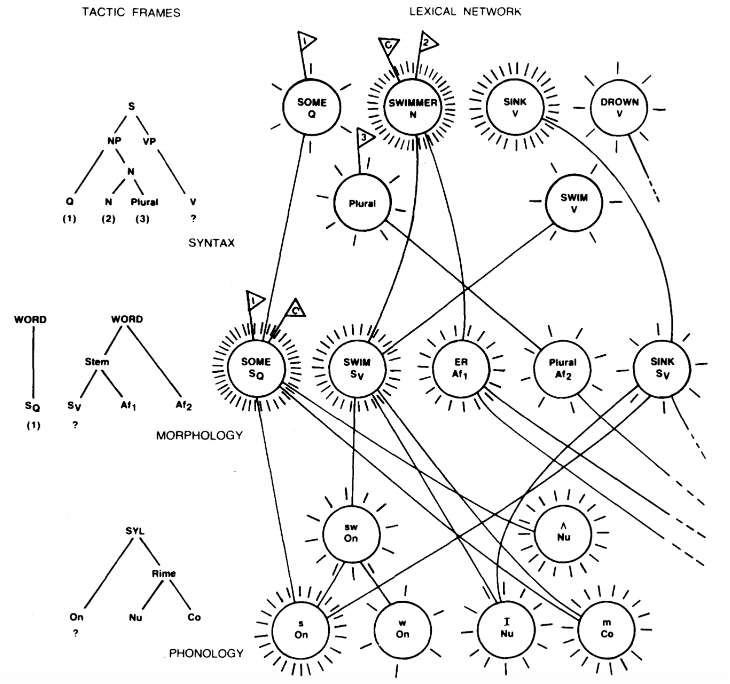

FIGURE 2.3: Production process for the sentence “Some swimmers sink”. From Dell (1986).

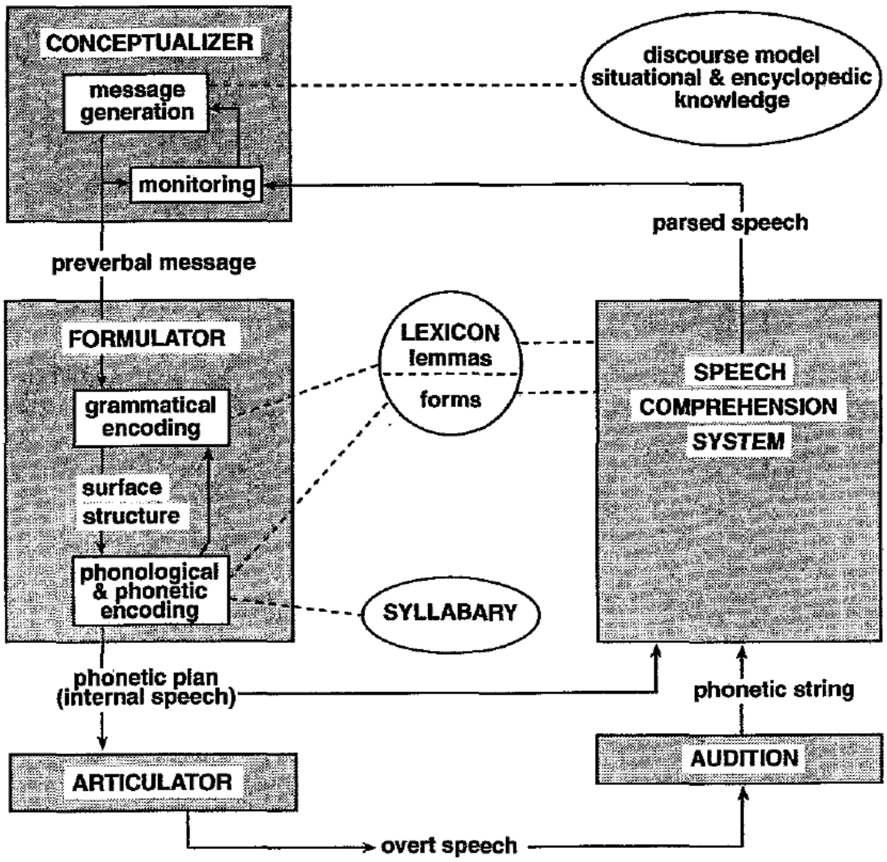

Figure 2.4: Levelt's model of speech production. From (Levelt, 1989, reprinted courtesy of The MIT Press)

model consists of three main autonomous components that are responsible for different aspects of speech production (see Figure 2.4):

1. Conceptualizer: It is responsible for generating the message to be communicated, encoding it into some kind of coherent plan, and monitor what is about to be said as well as what has been said. For the generation of a message, it has access to procedural and declarative knowledge, including encyclopedic knowledge of the speaker, knowledge about the situation, and the discourse record of the conversation. The message generation occurs in two stages: macroplanning and microplanning. The first consists of the elaboration of a communicative goal into a series of subgoals, and the retrieval of information to be expressed in order to realized these subgoals. Microplanning involves giving a propositional shape to each of these 'chunks', and assigning the informational perspective (topic and focus), that will guide the addressee's attention. The product of these two stages is a Preverbal Message, an organized conceptual structure that is not yet linguistic, and that constitutes the input for the next component.

2. Formulator: It is responsible for translating messages into a phonetic or articulatory plan. In order to achieve this, it has access to a mental lexicon, which is a repository of knowledge about the words in the language. Each lexical item is specified by a lemma, containing declarative semantic and syntactic knowledge, and a form, containing information about the morphology and phonology of the word. Formulation occurs also in two stages: grammatical and phonological encoding.

   During grammatical encoding, the lemmas whose meaning match part of the preverbal message are activated, which will make their syntax available, which in turn will activate certain syntactic building procedures. Using these procedures, the *Grammatical Encoder* builds syntactic structures such as verb or noun phrases. Retrieval of lemmas (lexical access) occurs following the Spreading Activation Theory (Dell, 1986), described above. In this view, the lemmas selected are those with the highest activation, which varies according to how their semantics match the concepts in the preverbal message. When the relevant lemmas have been retrieved and the related syntactic procedures have finished, the Grammatical Encoder would have produced a *surface structure*, an ordered string of lemmas, grouped in phrases and subphrases of various kinds.

   During phonological encoding, a phonetic or articulatory plan is retrieved and built for each lemma and for the entire utterance. This is done using the form information in the lexicon related to each lemma. Afterwards, several phonological procedures would modify the form information that is retrieved. The result of this process, the phonetic/articulatory plan (alternatively called "internal speech"), becomes the input for the next component.

3. Articulator: It specializes in the motor execution of the phonetic plan, involving the respiratory, laryngeal and supralaryngeal systems. Considering possible differences between the rate in which the Formulator delivers the phonetic plan, and its execution by the Articulator, an *Articulatory Buffer* is proposed, where the phonetic plan can be temporarily stored. The Articulator retrieves chunks from the Articulatory Buffer and unfolds them for execution. The product of this component is overt speech.

The monitoring process, for which the conceptualizer is responsible, is assumed to occur utilizing components of language comprehension. Thus, speakers can monitor their internal speech, detecting problems before articulation; and their overt speech, assessing the meaning and well-formedness of their productions.

The components are modular in the sense that each component depends only on its input and apart from that there is no communication or interaction with other components.

The process is incremental, as the output of each module is delivered as soon as the required information is available, and are normally used immediately by the next module. The model works both serially and in parallel, where the output of one module becomes the input of the next one, and all modules work at the same time although on different parts of the message. Finally, some parts of the model are assumed to work automatically without allocation of attentional resources. These aspects: incrementality, parallel and serial processing, and automaticity; allow for real time language production.

Although some of its parts and assumptions have been questioned, this model has been taken by many researchers as a framework for language production. It gives a complete view of the process, dividing it into several subtasks. Each one of the latter has been approached by different models, each one trying to describe and empirically verify their mechanics.

2.1.6 Aphasia Model (Dell et al., 1997)

Following the principles of the Spreading Activation Theory, Dell and colleagues proposed the Aphasia model (Dell et al., 1997). This model focuses on explaining errors concerning lexical access of aphasic and non-aphasic speakers in picture naming experiments.

The model describes how a pattern of activation corresponding to the meaning of a word is translated to a pattern corresponding to the sounds of the word. This process is assumed to occur in two steps:

1. Lemma Selection: a concept is mapped onto a lemma, a non-phonological representation of a word, which is often associated to grammatical properties such as gender and number.
2. Phonological Encoding: a lemma is transformed into an organized sequence of speech sounds.

Evidence for these steps comes from the tip-of-the-tongue (TOT) phenomenon, where a speaker is aware that a word exists but cannot access its sounds. Furthermore, some studies show that speakers in a TOT state know some grammatical properties of the word being sought, such as gender (Miozzo and Caramazza, 1997; Vigliocco et al., 1997). These steps are represented in the aphasia model (see Figure 2.5) by semantic features being mapped onto words and then onto phonemes.

Lemma selection starts by adding activation to the semantic features of the intended word. Then, activation spreads for a fixed number of time steps and according to a noisy

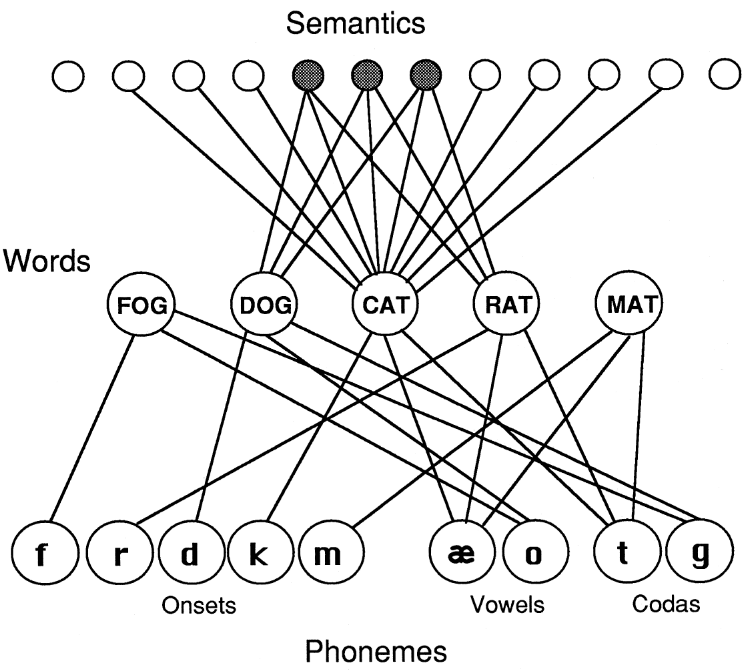

FIGURE 2.5: Aphasia Model. Connections are excitatory and bidirectional. From Dell et al. (1997).

linear activation update rule. Bidirectional excitatory connections of the model cause that the activation spreads in the 3 levels. At the end of this process, the most activated word of the correct syntactic category is chosen. Because of the activation noise, there is a small chance that a formal, semantic or mixed neighbor is chosen, instead of the intended word. It is also possible although more unlikely that a completely unrelated word is chosen.

Phonological encoding begins with a boost of activation of the word chosen during lemma selection. This boost introduces a nonlinearity that permits the model to handle the arbitrary mapping between semantic features and phonemes. Activation spreads then for another fixed number of time steps. At the end, the most highly activated phonemes are selected and linked to slots in a phonological frame. Errors at this point can occur if, due to noise, one or more phonemes are more active than those of the selected word. This would typically result in non-words.

This model was computationally implemented and used to explain the speech errors of aphasic and non-aphasic speakers. In order to accomplish that, the authors manipulated 2 variables: the connection weights between each level, and the decay rate of activation. In both cases, speech errors are caused due to a higher level of noise introduced. Nonetheless, lesions to connection weights promoted errors producing non-words or unrelated words; while a higher decay of activation was related to more formal, semantic or mixed errors, showing that even though the noise was dominating, the errors were related to the intended word because the connections between levels were still strong. Modifying only these 2 parameters, Dell et al. (1997) were able to fit the distributions of errors of 21 out of 23 aphasic patients, being able also to fit them after some recovery of the patients by manipulating the same parameters.

Being able to account for speech errors, permitting a graceful degradation through only altering a couple of parameters, the aphasia model was one of the most influential models of the speech error literature. Its key limitations, however, included that the connection weights were manually set and not learned, and that the output of the model was not sequential and phonemes were all retrieved at once. These two aspects were taken into consideration in the Phonological Error model (Dell et al., 1993), presented below.

2.1.7 Phonological Error Model (Dell et al., 1993)

While the aphasia model involved a mapping from a semantic representation to a word and then to phonemes, the Phonological Error model focuses on the process of Phonological Encoding, that is, on mapping a word to a sequence of phonemes. This model tries to integrate the principles of Parallel Distributed Processing (PDP), which stresses the parallel and distributed nature of (artificial) neural networks, and the process of learning the associated parameters.

This model implements a Simple Recurrent Neural network (SRN, Elman, 1990; Jordan, 1986) mapping a word representation to a sequence of phonological features (see Figure 2.6). The input layer ("Lexical" in Figure 2.6) contains a representation of the word to be spoken, that remains unchanged during the production of the word. In different versions this representation was either a random bit vector related to a lemma or semantic representation, or a vector correlated to the form of the word (either an underlying phonological representation or the orthographic input for a reading aloud task). The output layer contains 18 units corresponding to 18 phonological features. The input and output layers are connected via a hidden layer. Additionally, the model contains two sets of recurrent connections or context units feeding the hidden layer, one containing the activation of the hidden layer at the previous time step ("Internal Context" in Figure 2.6) and another one containing a copy of the output layer at the previous time step ("External Context" in Figure 2.6).

The word production process occurs as follows: at the beginning, the internal context units are set to zero, and the external context units are set to a pattern signaling a word boundary (0.5 in every unit). Production then starts when the input units are activated in a pattern corresponding to the target word. Activation spreads from input and context units to the hidden layer, which in turn spreads its activation to the output

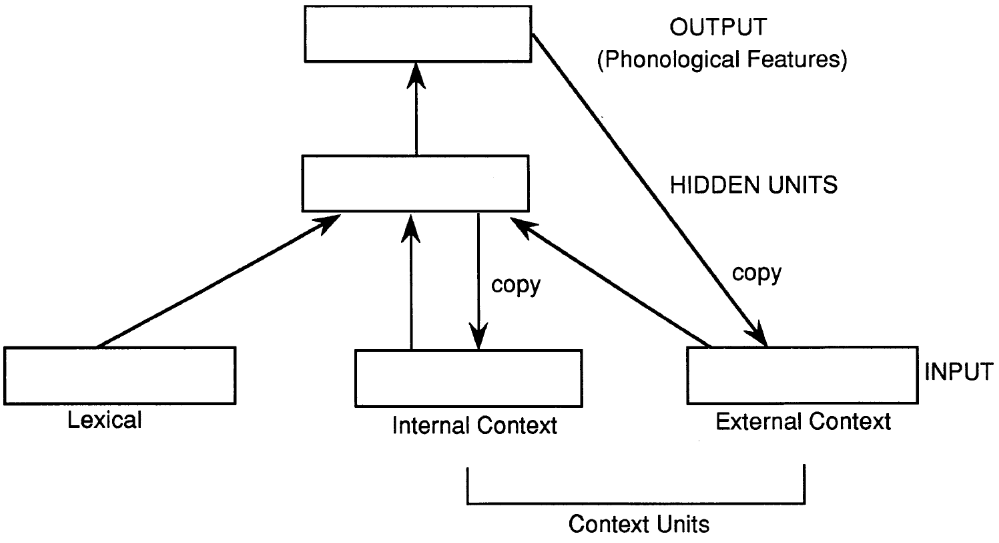

Figure 2.6: Phonological Error Model. Each rectangle represents a set of units. From Dell et al. (1999).

units. To the extent that the output differs from the expected output, the connection weights are changed using the backpropagation algorithm (Rumelhart et al., 1986), thus conflating training with processing. Then, the activation of the hidden units and output units are copied to the internal and external context units respectively. At the next and following time steps, changes in context units allow the model to produce the next elements in the sequence. Finally, production stops when a word boundary pattern is produced.

The focus of this model was to explain phonological speech errors giving an account that did not use a frame-and-slot approach. Frame-and-slot models assume two processes during word phonological encoding: one involving the retrieval of the sounds of a word, and another one involving the retrieval of a phonological frame. A frame represents the number of syllables and the location of stress in a word. Within a frame each syllable is associated with slots that label the kind of sounds acceptable for that slot. Some speech errors have been proposed as evidence for this approach (Shattuck-Hufnagel, 1979), including:

- Phonotactic regularity errors: speech errors tend to follow the phonotactic patterns of the language.
- Syllable constituent errors: syllables are thought to have an internal constituent structure, and speech errors reflect this structure. For example, considering a CVC syllable, one is more likely to elicit an error regarding either an onset (C) or a coda (VC) than other combinations.

- Sound exchanges: for example “heft lemisphere”, where an initial anticipatory substitution appears to cause another substitution in which the replaced sound replaces the anticipated sound; suggesting under a frame-and-slot approach that each sound was erroneously placed in the frame slot of the other word. Although these errors are not very common (only 5-10% of phonological errors), they are clearly not random events.

In order to produced speech errors, noise was introduced into the connection weights. The resulting errors had a strong tendency to follow the phonotactics of the language. Furthermore, errors tended to involve the hypothesized frame constituents, having more syllable onset (C) than syllable coda (VC) errors, and more rime (VC) than CV errors.

This behavior was attributed on the one hand, to the sequential nature of the architecture, where each item to be produced depends on the previous ones; and on the other hand, to the statistical nature of the corpus used to trained the model. Thus, errors reflect English patterns because the model was trained on English words, such that during training, weight changes created pathways describing possible derivation sequences, and when an error is elicited, the model sticks to those pathways. Moreover, errors related to a syllable constituent structure reflect statistical patterns of English, where at the beginning of a word there is more uncertainty than at the end, hence, there are more onset than coda errors; and more errors tend to involve rime (VC) than CV units because English has fewer VCs than CVs (Kessler and Treiman, 1997).

Although the model was able to account for a large proportion of the errors, it supplied no mechanism to explain sound exchanges. Nonetheless, its principles highlighted two important aspects during language production: its sequential and incremental nature, and the influence of statistical patterns in the corpus used to train the model.

2.1.8 Structural Priming Model (Chang et al., 1997)

This was the first computational model of human sentence production. It was developed to simulate structural priming, which is the tendency of speakers to repeat syntactic structures of sentences that were recently spoken or heard (Bock, 1986b; Bock and Loebell, 1990; Pickering and Ferreira, 2008). The central claim of this model is that structural priming is a form of implicit learning; thus, it is a consequence of the same mechanism through which the model learns to produce sentences.

The model follows connectionist principles and its architecture (see Figure 2.7) is similar to the Phonological Error model, where the input layer has learnable connections to a hidden layer, which in turn has learnable connections to an output layer. Additionally,

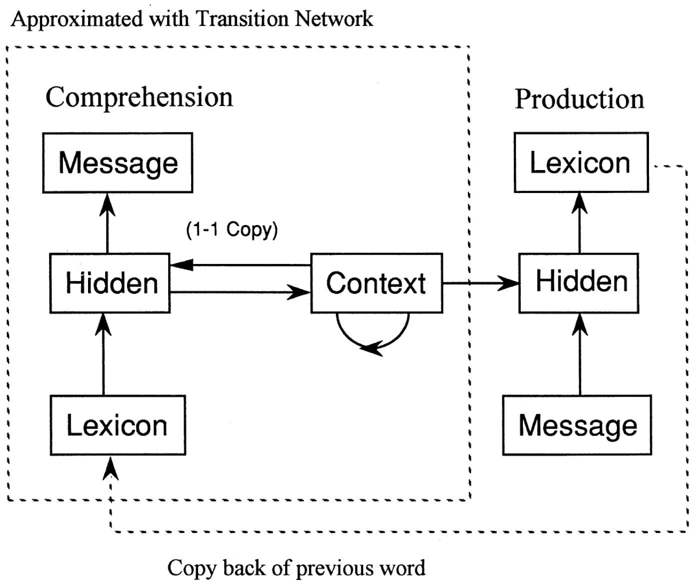

Figure 2.7: Structural Priming Model. Figure from Dell et al. (1999).

context units influence the activation of the hidden layer, however, in this case the context units are assumed to be the result of a comprehension system that is linked to production.

The input layer contains a representation of the message to be conveyed in the form of a 87-dimensional vector, where each dimension corresponds to a localist unit signaling semantic features. These features are grouped in blocks corresponding to event roles of the message: agent, patient, location and action. For example, the agent group contains features including CHILD, MALE and UNITARY. The representation of this message remains unchanged during the production of the sentence.

While the activation of the input units related to these features signaled its true value, the relative degree of activation signaled the aspects upon which sentence production should focus. Thus, in active sentences, the agent block would present relatively more activation than the patient block, and vice versa in passive sentences. Similarly, the difference between representations preferring a double object dative versus a prepositional dative was a difference in activation between the patient and recipient blocks. This is meant to reflect studies showing lexical priming effects on grammatical role assignments, where the easier it was to select a word to express a substantive concept, the more likely it was to be encoded as a sentential subject (Bock, 1986a, 1987). Furthermore, other studies showed that more conceptually available elements in the message (due to being more topical, imageable, animate or prototypical) are placed in more prominent grammatical roles than are less accessible ones (Bock and Warren, 1985; Ferreira, 1994).

Chang et al. (1997) made certain aspects of the meaning representation more accessible by assigning them relatively more activation.

The context units in this model are supposed to approximate input from the comprehension system which is assumed to be accessible during production. In this view, the context units, called *Transition Network*, reflect the current state of the sentence from the perspective of the comprehension system. This information is represented by 10 units fully connected to the hidden layer, each one associated to a syntactic or an event role category (e.g., VERB, AUX, AGENT, PATIENT). During production, activation of these units vary according to the previously produced word. For example, after a noun, the AGENT and PATIENT units would become active, reflecting that the previously produced noun could be either an agent or a patient with some ambiguity; meanwhile, after a verb, the VERB unit would become active, in this case without ambiguity. Additionally, each node retained half of its activation across the production of each individual word.

The output layer contains 59 localist units corresponding to words in the vocabulary. The training corpus consisted of 3600 sentences including different syntactic alternations, such as passive or active sentences, and double object or prepositional dative.

The process of sentence production is as follows: The context units are initialized with a PERIOD (which is different from the period symbol in the output layer), signaling the beginning of a sentence. The input units are activated to the pattern corresponding to the message to be conveyed. At each time step, activation propagates from input and context units to the hidden layer, which propagates its activation to the output layer. The word with highest activation at the output layer is the one selected for production. Then, activation of the context units change as described above, for example, after a verb is produced, the context unit VERB would be activated, while the context units activated at a time step before would reduce its activation by half. Similar to the Phonological Error model, the input units remain with a constant activation during the whole process, while the context units change their activation according to the state of production, allowing the model to have a sequential behavior.

Training was performed using the backpropagation algorithm (Rumelhart et al., 1986) with weight changes after each word, with a learning rate of 0.06 for the first quarter of training and 0.03 for the rest, and with a momentum of 0.9. During training, each sentence in the corpus was shown to the model an average of 31 times. The model was tested afterwards on 400 sentences, out of which 74% were novel sentences. On this set, the model produced the correct word 94% of the time.

In order to test structural priming, the learning mechanism continued working. First, the model was shown a prime message that required either an active or a passive sentence (alternatively a double object or prepositional dative). As in training, the connection weights were adjusted as the sentence was produced with a learning rate of 0.03. Finally, the model was given a target message that was neutral with respect to conceptual accessibility; for example, a message that could be encoded by a passive or active sentence would have equal activation in the agent and patient blocks. The percentage of times that each structure was produced as a function of the prime was recorded, the differences constituted a measure of the priming effect.

The sizes of the priming effects patterned with the data. For example, having as prime "boys chase dog", the model would promote "girls feed cat" over "cat is fed by girls". Furthermore, the model successfully simulates the persistence of priming over 10 intervening sentences (Bock et al., 1996), exhibiting the phenomenon that motivated the view of priming as implicit learning. Nonetheless, the model failed to exhibit some of the effects shown by Bock and Loebell (1990), where the priming seemed to be related to the surface constituent structure of the sentence, rather than to a mapping between event and grammatical roles. The reason for this was proposed to be at least partially due to the nature of the input representations, where each feature in the message is completely independent of the others, denying any possible similarity.

2.1.9 Dual-Path Model (Chang, 2002)

The evidence of Bock and Loebell (1990) suggested that some structural priming effects were related to promoting the surface constituent representation of the prime sentence, rather than relations between specific semantic concepts and grammatical roles. This would entail that between semantics and syntax, there is a significant independence or abstraction, where a grammatical construction could be promoted in a general way and not just for the specific semantic concept related to the priming sentence.

This dissociation between syntax and semantics was taken by Chang (2002) as inspiration for his Dual-Path model, which has since become the most influential connectionist model of sentence production. In this view, event roles and semantic concepts can be bound in a manner that does not necessarily reflect the training experience, permitting bindings that were not seen before during training. This would allow the model to be able to generalize to a greater degree, showing *systematicity*, which refers to the ability of cognitive systems to handle novel elements generalizing from known ones. This ability is argued to be necessary while modeling human cognition in general (Fodor and Pylyshyn, 1988; Marcus, 1998a,b), and particularly while modeling human sentence production, as

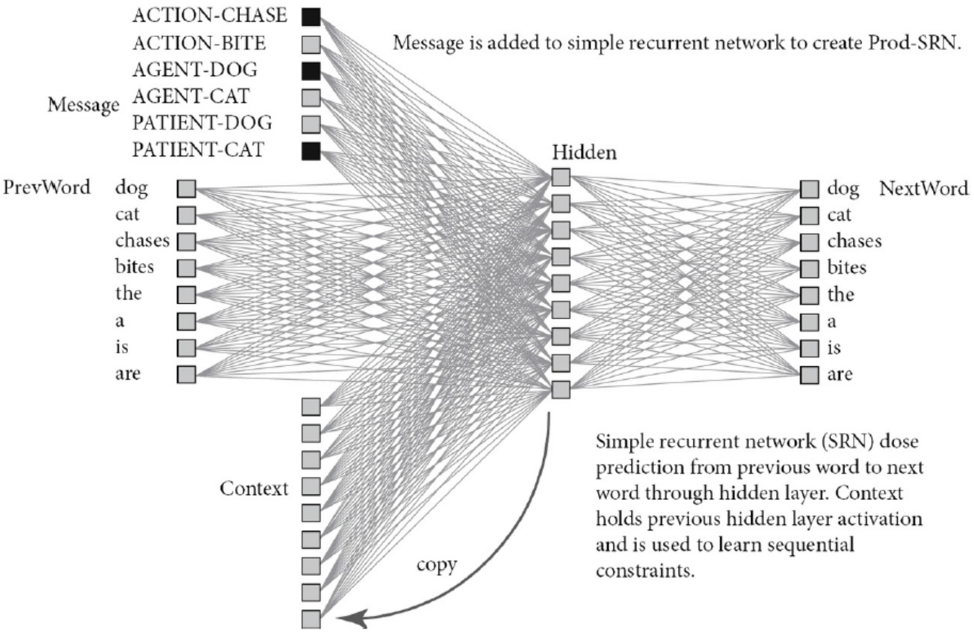
ACTION-CHASE
ACTION-BITE
AGENT-DOG
Message
AGENT-CAT
PATIENT-DOG
PATIENT-CAT
Message is added to simple recurrent network to create Prod-SRN.
Hidden
PrevWord
dog
cat
chases
bites
the
a
is
are
NextWord
dog
cat
chases
bites
the
a
is
are
Context
copy
Simple recurrent network (SRN) dose prediction from previous word to next word through hidden layer. Context holds previous hidden layer activation and is used to learn sequential constraints.

Figure 2.8: Production Simple Recurrent Network Model. From Chang and Fitz (2014).

the number of sentences in a language is infinite, and therefore most of them would be novel to a model (see also Dell et al., 1999).

Chang (2002) compares two architectures for sentence production: one that is very similar to the Structural Priming model, named Production Simple Recurrent Network (Prod-SRN, see Figure 2.8), and the Dual-Path model (see Figure 2.9). The focus of the comparison was to see how each architecture was able to generalize and produce novel sentences.

The Prod-SRN model utilizes the same type of input representations as the Structural Priming model. Furthermore, it also maps the input layer to a hidden layer and then to an output layer that contains the words in the vocabulary. However, the context units differ, as in this case they consist of two sets of units: one containing a copy of the activation of the hidden layer at the previous time step, and another one containing the word that was produced also at the previous time step. Compared to the Phonological Error model, the Prod-SRN model has the same architecture, but operates at the sentence level, producing sequences of words, instead of sequences of phonemes.

An important aspect of the message representations used by the Prod-SRN model and the Structural Priming model is that they are binding-by-space representations, meaning that if a concept appears in different roles, each combination concept-role would be represented by different neurons. For example AGENT-DOG and PATIENT-DOG would be represented by different neurons with no apparent relationship, even though they share the element DOG. Although this is a common practice in neural networks (e.g., Miikkulainen, 1996; St John and McClelland, 1990), this prevents the model from learning similarities and generalizations over slots and fillers.

The Dual-Path model, shown in Figure 2.9, presents a more complex architecture compared to the previous models. The key feature of this model is its role-concept binding mechanism (see top right of Figure 2.9), which in contrast to binding-by-space representations where for example the pair AGENT-DOG would be represented by a localist unit, the Dual-Path model introduces a *binding-by-weight* mechanism, which is implemented by temporal links within the architecture. For example, considering the roles AGENT, PATIENT and ACTION, and the concepts DOG, CAT, CHASE and BITE; the message of "dog chases cat" would be represented in the network by having a constant positive connection weight from the role AGENT to the concept DOG, from PATIENT to CAT, and from ACTION to CHASE (as appears in Figure 2.9) and all other connections would have connection weights of zero. In this way, activation can flow from the role AGENT to DOG, but not to CAT, as the latter has no connection to AGENT in this given example. Similarly, for the sentence "cat bites dog", there would be positive connection weights between AGENT and CAT, PATIENT and DOG, and ACTION and BITE, and all other connections would have weights of zero. Thus, the connection weights in this part of the model are not learnable and constitute part of the message representation of the sentence to be produced. This type of binding is assumed to be related to the spatial processing mechanisms of the brain, in which object and location/action information are represented in different pathways, which are then bound together for spatial processing (Goodale and Milner, 1992; Mishkin and Ungerleider, 1982).

At the core of the architecture, a hidden recurrent layer is responsible for the sequential nature of the model, this layer is associated to a set of context units that contain the activation of the hidden layer at the previous time step. Additionally, one can distinguish two pathways of processing. The first one, named the *sequencing system*, corresponds to the pathway PrevWord > Compress > Hidden > Compress > NextWord in Figure 2.9. The layer PrevWord consists of localist units where only the word produced at the previous time step is active. The NextWord layer corresponds to the output layer and it also consists of localist units where each unit represents one word in the vocabulary. The Compress layers represent steps of abstraction, the first one maps a word to an abstract low-dimensional representation, proposed to be similar to syntactic categories; while the second one maps an abstract representation to specific words. As one can see, this pathway is very similar to the segment PrevWord > Hidden > NextWord of the Prod-SRN model (see Figure 2.8), with the exception of the two Compress layers. According to the author, this pathway learns sequential behavior of words, as it has

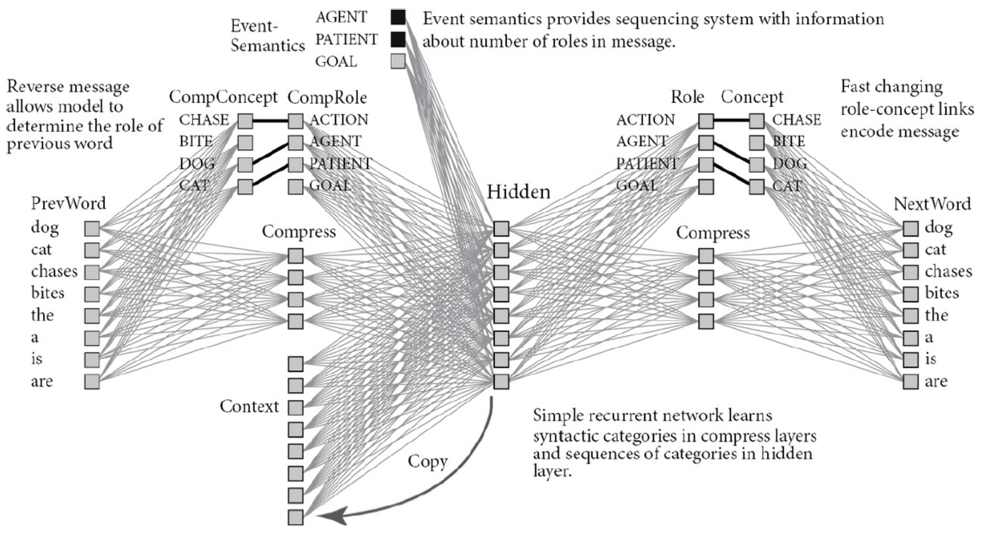
Event-
Semantics
AGENT
PATIENT
GOAL
Event semantics provides sequencing system with information about number of roles in message.
Reverse message allows model to determine the role of previous word
CompConcept
CompRole
CHASE
BITE
DOG
CAT
ACTION
AGENT
PATIENT
GOAL
Role
Concept
ACTION
AGENT
PATIENT
GOAL
CHASE
BITE
DOG
CAT
Fast changing role-concept links encode message
Hidden
PrevWord
dog
cat
chases
bites
the
a
is
are
Compress
Compress
NextWord
dog
cat
chases
bites
the
a
is
are
Context
Copy
Simple recurrent network learns syntactic categories in compress layers and sequences of categories in hidden layer.

Figure 2.9: Dual-Path Model. From Chang and Fitz (2014).

semi-direct connections to the next and previous words (only divided by the Compress layers).

The second pathway corresponds to PrevWord > CompConcept > CompRole > Hidden > Role > Concept > NextWord. The segments CompConcept > CompRole and Role > Concept correspond to the binding mechanism previously described and their connections are defined by the message to be conveyed. The segment PrevWord > CompConcept > CompRole can be thought as a comprehension counterpart to the segment Role > Concept > NextWord. In this case, PrevWord contains the word produced at the previous time step which learns to activate its related concept, which in turn activates the appropriate role given the temporary links, which effectively feeds the hidden recurrent layer with the role activated at the previous time step. In this way, one can see that the hidden recurrent layer does not operate directly over concepts in the message representation but only over event roles. Consequently, the syntactic behavior of the recurrence is independent of lexical content.

The last part of the model consists of a block of localist units called *Event Semantics* (see top of Figure 2.9) signaling the number and type of roles to be encoded in the sentence. For example, one role for “the dog sleeps” and two for “the girl chased the boy”. Additionally, activation of this units varied systematically according to the intended sentence. For example, AGENT would be less activated if a passive sentence is to be produced. In general, this layer is useful for the model to select the proper structure when several alternatives are available.

Production occurs as follows: First the context units are initialized to 0.5, the event semantics units are set according to the type of sentence to be produced, and the connection weights between Role and Concept, and between CompConcept and CompRole are set according to the message to be conveyed. At the first time step, activation spreads from the context and the event semantics units to the hidden layer, which in turn spreads its activation both to the second Compress layer and to the Role layer. The Role layer would then activate the Concept layer. Finally, the output layer NextWord would receive activation both from the Concept layer and the Compress layer. The unit in NextWord with highest activation is selected as the word produced at that time step. The unit in PrevWord related to the previously produced word is activated, additionally the activation of the hidden layer is copied to the context units. At the following time steps, the process is identical except that the hidden layer also receives activation originated from PrevWord, traversing on one pathway the first Compress layer, and on the other pathway CompConcept and CompRole. This continues until the model produces a period ".", signaling the end of production.

These two architectures (the Prod-SRN model and the Dual-Path model) were both trained using backpropagation (Rumelhart et al., 1986). After training they were tested for generalization according to 3 tasks. In the first task the models were tested to see if a word could appear in a role that was not seen during training; for example, if the models were able to produce "dog" in a goal position of a dative sentence. The second task tested whether the models were able to produce structures of the form "a X is a X", in which the models had to generalize a novel word to 2 sentence positions; for example "a blicket is a blicket" (Marcus, 1998b). The third task tested if the models were able to produce novel adjective-noun pairs; for example, produce "happy cake", where during training "happy" was only seen coupled with animate nouns and during testing it was tested whether it could be produced modifying inanimate nouns.

In all 3 tasks, the Dual-Path model showed considerably greater degrees of generalization, providing a closer match to human level syntactic behavior. For instance, for the first task the Dual-Path model produced correct utterances 88% of the time, while the Prod-SRN model only 6%. The overall difference was attributed to the fact that the syntactic knowledge in the Dual-Path model was separated from the lexical content of the message. This separation was needed to ensure that abstract syntax was learned. As explained by Chang (2002), the Prod-SRN model showed poor generalization, and rather memorized the mapping between particular meanings and particular word sequences, partly due to the binding-by-space localist representations, and partly due to its architecture. Consequently, the Prod-SRN model violated the property of Systematicity, which Fodor and Pylyshyn (1988) argued is a fundamental feature of human cognition.

The Dual-Path model was further tested on a variety of structural priming phenomena in Chang et al. (2006), assuming that priming was a product of implicit learning. Similar to the Structural Priming model, it was more likely to describe a target message using the same structure of the prime, regardless of the number of fillers intervening between prime and target, reproducing the effects of Bock and Griffin (2000). Furthermore, the model successfully reproduced the effects of Bock and Loebell (1990) and Bock (1989), where priming was related to the surface structural representation of the prime, rather than to its semantic content or to specific function words. Nonetheless, according to Chang et al. (2006), due to the small learning rate, inherent to the account of priming as implicit learning, the Dual-Path model was unable to show *lexical boost*, which is the strong but short-lived priming effect that occurs when the prime and target present overlap of content words such as verbs or nouns (Cleland and Pickering, 2003; Hartsuiker et al., 2008; Pickering and Branigan, 1998). Consequently, Chang et al. (2006) suggested that lexical boost was caused by a different short-lived mechanism, a prediction that was confirmed two years later by Hartsuiker et al. (2008).

Additionally, the Dual-Path model has been tested comparing a Japanese and an English version (Chang, 2009). The model was able to learn the patterns of each language with similar levels of grammaticality while producing novel sentences (93% for English and 95% for Japanese). Furthermore, it could explain production preferences between speakers of the different languages, namely, Heavy NP shift (Arnold et al., 2000; Hawkins, 1994, 2004; Ross, 1967) and aspects regarding lexical/conceptual accessibility. Heavy NP shift refers to the tendency of English speakers to prefer configurations where long phrases are placed later in sentences, showing a short-before-long bias. Japanese, on the contrary, presents a long-before-short bias (Hawkins, 1994; Yamashita and Chang, 2001). These preferences were reproduced by the model, and an analysis of its internal representations suggested that the phenomenon was caused by a difference in the relative importance of meaning and surface structural information at the choice point where the two word orders diverge. In English, this choice point is placed after the verb, while in Japanese it is at the beginning of the sentence.

Regarding lexical/conceptual accessibility, in English speakers prefer to use animate elements early in a sentence (McDonald et al., 1993) which can lead to the use of less common structures, like passives (e.g. "the man was almost hit by a car"). Using the Dual-Path model, Chang (2009) noticed that these preferences were related to the frequency of the input sentences, where sentences with animate subjects were more frequent, and after training, the model would present this preference as well. Analogous patterns were reproduced in the Japanese model, showing that while both versions shared the same architecture, each one adapted to the statistical regularities of its language.

Other phenomena in which the Dual-Path model was tested include preferential looking during language acquisition (Chang et al., 2006), the production of relative clauses during language acquisition (Fitz, 2009) and aphasic sentence production (Chang, 2002; Gordon and Dell, 2003).

2.2 Uniform Information Density Hypothesis

In the previous section, some of the main models of human language production were introduced. They described the process of production by defining representations and operations over these representations. Thus, they can be regarded as operating at the algorithmic level of analysis (Marr, 1982). In this section we will describe the *Uniform Information Density Hypothesis* (UID, Jaeger, 2006, 2010; Levy and Jaeger, 2007), which is an account of language production operating at the computational level of analysis, defining the goal of the computation and why it is appropriate.

UID stems from the notion that human language production is an efficient process, which has been approached before. Considering the relation between word frequency and form, Zipf (1929) noticed that frequent words have shorter forms, and later proposed the Principle of Least Effort (Zipf, 1949), stating that human behavior presents a preference to minimize the amount of work over time. More recently, Piantadosi et al. (2011) showed that word length is even more strongly correlated with the average word predictability of a word in its context than with its frequency alone. Furthermore, it has been found that more predictable instances of the same word have on average shorter durations and less phonological/phonetic detail (e.g., Aylett and Turk, 2004, 2006; Bell et al., 2003, 2009). This relation reflects information theoretic considerations about efficient communication (Shannon, 1948), where words that add more information to their context (are less predictable) have longer forms.

UID builds on the *Entropy Rate Constancy Principle*, which states that the entropy related to sentences remains constant through a text. Then, if one does not consider context, the entropy of a sentence would increase with the location of the sentence within the document (Genzel and Charniak, 2002). Similarly, the *Smooth Signal Redundancy Hypothesis*, proposes that the pressure of producing robust communication while efficiently expending articulatory effort leads to an inverse relationship between language redundancy and duration. Such behavior would improve robustness in communication by spreading information more evenly across the speech signal (Aylett and Turk, 2004). In both cases, the speaker is assumed to behave rationally (see Anderson, 1991), trying to optimize its behavior in order to achieve optimal communication.

Expanding from these notions, UID proposes that language production is affected by a preference to distribute information across the linguistic signal as uniformly as possible given the encoding possibilities of the language. This assumes that human communication is efficient, balancing the risk of transmitting too much information per time, increasing the risk of information loss or miscommunication; and the desire to convey as much information as possible with the least amount of linguistic units.

Following such strategy, speakers would achieve optimal information transmission over a bandwidth-limited noisy communication channel (Genzel and Charniak, 2002), by transmitting a constant amount of information per time, and that is close to, but below, the channel capacity. The channel capacity is defined as the maximum amount of information per transmission through a noisy channel that can be transmitted with an arbitrary small error rate (Shannon, 1948), and thus characterises the upper bound of the processing capacity of the comprehender.

Additionally, following UID, speakers would minimize the comprehension effort related to the produced utterances, provided that the effect of surprisal on comprehension difficulty is superlinear (Levy and Jaeger, 2007); where surprisal refers to the negative log probability of a linguistic unit conditioned on its context, which measures the amount of information contained in that unit:

$$surp(x) = -logP(x|context) \tag{2.1}$$

The way to achieve uniform surprisal profiles (information density) relies on the availability of multiple ways of encoding a meaning representation into linguistic units, which depend on the constraints given by the grammar of the language. According to UID, a speaker would prefer among these possibilities, the ones with more uniform surprisal profiles.

For example, in the case of that-ommission in complement clauses (see Figure 2.10), speakers have the option of including the complementizer “that” or not. The channel capacity in Figure 2.10 is represented by the gray horizontal line in each graphic. Considering word surprisal and how it evolves over time, in the first case (Figure 2.10, above) omitting the complementizer would result in surprisal at “we” that would be beyond the channel capacity, which could cause comprehension/communication errors; therefore, mentioning the complementizer would be preferred, decreasing the information density at that point. For the second case (Figure 2.10, below), omitting the complementizer renders a sentence that is already below the channel capacity, mentioning the complementizer would decrease the information density even more, which would reduce the

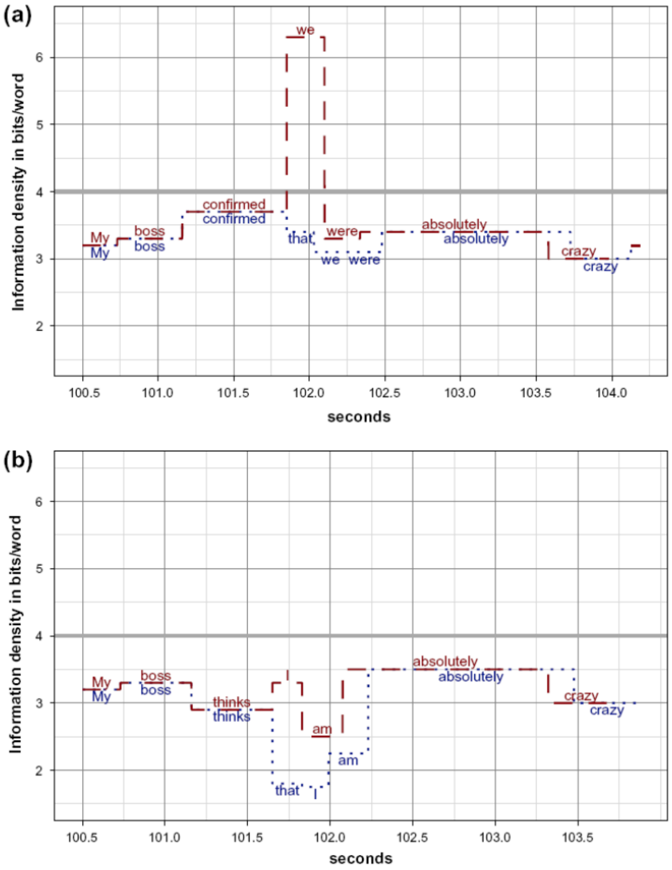

Figure 2.10: Illustration of the development of information density over time for two alternative ways to encode the same message. A purely hypothetical channel capacity is indicated by the solid horizontal line. From Jaeger (2010).

efficiency of communication; consequently, in such case, omitting the complementizer would be preferred.

Evidence supporting this hypothesis has been found at different linguistic levels. Concerning speech rate, words with high information content are spread over longer periods of time (Aylett and Turk, 2004; Bell et al., 2003). Similarly, phonemes that are highly informative are produced more slowly and with more articulatory detail (Van Son and Van Santen, 2005). At the morphology level, Frank and Jaeger (2008) showed that full forms of “be”, “have” and “not” tend to be used at points of high information density, as opposed to reduced forms (e.g., “I am” vs “I’m”). At the level of syntax, speakers are more likely to produce optional functional words (“that”), when the following words would otherwise be high in information content (Jaeger, 2006; Levy and Jaeger, 2007).

While evidence suggests the plausibility of UID, some important issues remain. One is the study of the nature of the channel capacity, which is hypothesized to bound communication, but it is not clear what factors affect it, or even how to measure it. Another issue concerns a description at the algorithmic level of analysis. The UID hypothesis only describes why a uniform information transmission rate is desirable, and the empirical evidence supports the idea of its existence; however it is not clear what mechanism would implement such strategy.

2.3 Summary

This section summarizes the main points presented in the previous two sections, enumerating aspects that are important to be considered while modeling language production, and highlighting those that are most relevant for this thesis:

- Speech errors suggest that language production is performed through a sequence of levels. The units over which each level operates are organized such that during production, similar elements compete, and errors are elicited when a similar but different element is activated in exchange of the intended one.
- Lexical bias and the overrepresentation of errors involving both form and meaning similarity suggest that different levels interact both top-down and bottom-up.
- The Aphasia model Dell et al. (1997) demonstrates that some speech errors can be explained by varying the level of noise within a neural network. Additionally, it corroborated that an architecture with top-down and bottom-up interactions is able to explain the phenomena mentioned in the previous point.
- The Phonological Error model (Dell et al., 1993) shows that a large proportion of errors can be accounted by implementing a sequential and incremental architecture that creates dependencies and coherence amongst the elements produced. Additionally, it highlighted the importance of the statistical patterns of language, being able to explain different phenomena during phonological encoding.
- The Structural Priming model (Chang et al., 1997) shows that a similar architecture as the Phonological Error model could be used to explain phenomena at a sentence level. Specifically, it was able to explain structural priming as a product of implicit learning.
- Systematicity is an important attribute of cognitive models, and specially for sentence production, since the number of sentences related to a language is infinite. The Dual-Path model was able to produce sentences with high levels of systematicity by separating syntax from semantics, which was achieved by a binding-by-weight mechanism that merges the architecture of the model with the message representation. This same mechanism was able to reproduce the effects of Bock and

Loebell (1990), which also suggested a separation between syntax and semantics, amongst other phenomena.

- As explained by Chang et al. (1997) and Chang (2002), a lack of systematicity is at least partially due to the nature of localist representations, where similarity between different elements cannot be assessed by looking at their representations.
- The Uniform Information Density Hypothesis Jaeger (2006, 2010); Levy and Jaeger (2007) is an account placed at the computational level of analysis that considers language as an efficient process, reflecting information theoretic considerations about efficient communication. In such account, linguistic units are produced with a preference to distribute information across the linguistic signal as uniformly as possible given the encoding possibilities of the language, balancing, on the one hand, the risk of transmitting too much information per time, increasing the risk of miscommunication; and on the other hand, the desire to convey as much information as possible with the least amount of linguistic units. Evidence for this is available at different linguistic levels (e.g., Aylett and Turk, 2004; Bell et al., 2003; Jaeger, 2006), however, it is not clear what algorithmic mechanism would implement such account, or the way to measure the capacity of the communication channel.

The following chapters approach some of these aspects by presenting a new model of sentence production. This model possesses a similar architecture to the Phonological Error model and the Structural Priming model. Consequently, this new model is also incremental, sequential and able to learn and behave according to the statistical patterns of the training examples. Nonetheless, the semantic representations that are used as input are distributed and continuous, and thus to some extent mitigate the difficulties associated to the use of localist symbolic representations. Using these continuous representations, the model is tested for systematicity, demonstrating a high performance, while reproducing some findings of the speech error literature. Additionally, the internal dynamics of said model is analyzed, arriving to an algorithmic account of language production using recurrent neural networks. Finally, the model is extended in order to provide a first mechanistic account at the algorithmic level of analysis of the UID Hypothesis.

Chapter 3

Semantic Representations

The first stage of language production is related to the instantiation of a mental representation of the message to be conveyed. While the model proposed here is not meant to simulate the construction process of these representations, they are important as they constitute the input of the model, which fundamentally influences its behavior. If a model uses representations that are not appropriate, the performance of the model might be hindered, even if the architecture would otherwise permit a good fit to experimental data. Furthermore, if a model uses representations that are dissimilar to those used by humans, the explanatory value of the results might as well be questioned. In this perspective, identifying appropriate representations is a fundamental aspect to take into account during computational modeling, perhaps as important as the architecture.

For the simulations described here, the message to be conveyed corresponds to semantic representations based on the Distributed Situation Space model (DSS, Frank et al., 2003, 2009). Briefly put, each semantic representation under this scheme is a vector in a continuous multidimensional space, containing rich information describing the situations in which the corresponding sentence is true.

These representations were successfully used in the connectionist comprehension model described by Frank et al. (2009) (see also Venhuizen et al., 2018), showing that their model was not only able to comprehend sentences that it had seen during training, but that it is also able to comprehend sentences and situations that it had never seen before, thus showing semantic systematicity. As mentioned in the previous chapter, systematicity has been argued to be a fundamental property of cognitive systems (Fodor and Pylyshyn, 1988), and is specially important for sentence comprehension and production, considering that while the number of words in the vocabulary of a language is finite, the number of sentences allowed for the same language is infinite. As a result, any model trying to learn a language has the challenge of learning it without having access to all the possible sentences of it; and likewise, any model of language comprehension and/or production must be able to process sentences that have not been encountered before.

More concretely, for the simulations presented in the next chapters, two types of representations were used. The first one corresponds to the representations defined by Frank et al. (2009), called *situation vectors*. The second type is a modification of the latter which is introduced as part of this thesis, called *belief vectors*, which showed a better performance in the language production task. This chapter presents an overview of the Distributed Situation Space and its situation vectors, as well as the belief vectors, showing advantages and disadvantages of each type of representation.

3.1 Symbolic Localist Representations vs Situation Models

As seen with the models presented in the previous chapter, most connectionist models of sentence processing, either of production or comprehension, use symbolic localist representations, where only one or few dimensions of a vector are activated (set to 1), while the rest are set to zero; corresponding to a structural combination of symbols representing the propositional content of a sentence (e.g., Budiu and Anderson, 2004; Chang, 2002; Chang et al., 1997; Desai, 2007). While this is a simple way to introduce information into the models, the format of these representations might lack the expressiveness necessary to emulate the kind of mental representations that are utilized by humans during language processing.

Some studies suggest that during language comprehension, the semantic representation formed by the comprehender is not just a propositional structure of the utterance, as traditionally assumed (e.g., Kintsch and Van Dijk, 1978), but involves a simulation of the situation that the utterance describes, a *situation model*. This is similar to the theory of Johnson-Laird (1983), which states that the mental representation of the meaning of a proposition involves a representation of one or more concrete situations that are consistent with the proposition. Experimental evidence supporting this idea has been presented (for a review, see Kerkhofs and Haselager, 2006). For example, Stanfield and Zwaan (2001) found that readers mentally represent the orientation of objects when they are implied. Similarly, Zwaan et al. (2002) found that the shape of objects forms part of the mental representation of a sentence even if this shape is not explicitly mentioned or relevant.

Considering language comprehension as the construction of a simulation, the process would depend strongly on the experience and knowledge of the comprehender about the world (Frank et al., 2009). As a consequence, the mental representations used by computational models should reflect and include world knowledge about the implications of a given utterance.

In this context, Frank et al. (2009) further proposed that mental representations of this nature lead to *direct inference*. A representation of an event A is said to have the property of direct inference if the representation of A also represents any event B, if B is an implication of A (see also Haugeland, 1987). Consequently, relations between events in the world are reflected in the relations between the representations of the events. In this case, one can also say that the form of a representation is *analogous* to its meaning (Frank et al., 2009). Representations that are analogous and modal were referred to as "perceptual symbols" by Barsalou (1999), however, following Frank et al. (2009), we will use the term "symbol" in the sense of Peirce (1903/1985), referring to tokens with an arbitrary relation between form and meaning.

Symbolic localist representations contain very limited information of what is being represented. For example, a localist unit related to the proposition *play(charlie)* gives no information about its implications in the world, as in itself the representation is only an index that is related to a set of connection weights that link that representation to the rest of the network. The nature of what is represented has to be learned through training and the result of that training is the modification of the connection weights and not the representation. If after training one introduces a new unit *play(heidi)*, the model would have no means to encode the similarity of these two units, as they would be linked to two different sets of connection weights, and any relation would have to be learned by exposing the model to multiple training items involving both of these units. Consequently, the model would be unable to handle representations that have not been seen during training, exhibiting a lack of systematicity.

Another disadvantage of using discreet symbolic representations is that their representational space is finite and bound by the number of units. With pure localist representations where only one unit can be active at a time, the number of representable entities is equal to the number of units n. If multiple units can be active simultaneously, then the number of representable entities is at most 2^n. Even if 2^n grows rapidly with the number of units, it is still finite, which is problematic, considering that, because of recursivity, the number of possible sentences in a language is infinite.

In sum, the form of a symbolic representation provides no information about the nature of what is being represented (in this case, the situation models related to the meaning of sentences), since the form of each representation is not related to its meaning. Consequently, information about how a represented element relates to other elements cannot be obtained from the representation itself, and would have to be learned during training, preventing the model from properly processing elements that have not been encountered before. Additionally, the limited representation capacity of localist representations implies that they cannot be used to represent infinite sets, which is something necessary to represent the sentences of a language and the situation models that are related to them. These aspects highlight the need for an alternative representational scheme.

3.2 Distributed Situation Space Model

Taking into account some of the considerations presented in the previous section, the Distributed Situation Space model (DSS, Frank and Haselager, 2006; Frank et al., 2003, 2009) was proposed. The main motivation was to build semantic representations with the property of direct inference. This would be accomplished if co-occurrence relations amongst events would be apparent by comparing their representations. In particular, having only the representations of any pair of events: A and B, one should be able to calculate P(A|B). If this holds, the representation of B would also be representing anything that depends on B, thus, showing direct inference.

The representations obtained from the DSS model indeed have te property of direct inference. They are also analogous, as the form of each representation depends on what is being represented. They provide rich information about the meaning of each sentence in terms of the situations to which a sentence refers, permitting the comparison of any pair of semantics just by looking at their representations. Furthermore, they are continuous representations, and as such, the number of semantics that one can represent is potentially infinite, an attribute that is necessary in order to represent infinite sets, such as the set of sentences of a language. These aspects make DSS representations a possible solution to the disadvantages of symbolic localist representations that were described in the previous section.

More concretely, the DSS model represents events with respect to a *microworld*, which consists of a small set of entities interacting with each other, and which is structured in the sense that there are probabilistic constraints on event co-occurrence. This microworld is defined by a finite set of *basic events* (e.g., $play(charlie, chess)$ and $place(heidi, bedroom)$)—the smallest meaning-discerning units of propositional meaning in that world. Paired to a microworld, a *microlanguage* permits the generation of sentences expressing information about situations in the microworld. With these elements, pairs of the form (sentence,semantics) can be obtained, corresponding to the (input,output) of the comprehension model of Frank et al. (2009), and the (output,input) of the language production model presented in this thesis.

Class	Variable	Class members (concepts)	#
People	p	charlie, heidi, sophia	3
Games	g	chess, hide&seek, soccer	3
Toys	t	puzzle, ball, doll	3
Places	x	bathroom, bedroom, playground, street	4
Manners of Playing	m_{play}	well, badly	2
Manners of winning	m_{win}	easily, difficultly	2
Predicates	-	play, win, lose, place, manner	5

Table 3.1: Concepts in the microworld (Frank et al., 2009).

In the rest of this section, we will overview the microworld and the microwlanguage that were used for the simulations, as well as the process of obtaining the DSS semantic representations that have the properties mentioned above.

3.2.1 Frank et al. (2009)'s Microworld

For the simulations presented in this thesis, the same microworld defined by Frank et al. (2009) was used. In this microworld there are 3 people (2 girls and 1 boy), 4 places, 3 games and 3 toys, as shown in Table 3.1. People can be located at any time in one of the 4 places, and play a game or with a toy. Additionally, there are 2 manners of playing and 2 manners of winning. By combining each of the 5 predicates with their possible arguments, 44 basic events can be constructed (e.g. $play(charlie, chess)$, $win(sophia)$, $place(heidi, bathroom)$). These 44 basic events can fully describe the state of affairs of the microworld at any given point.

This microworld presents structure in the sense that there are probabilistic constraints on event co-occurence. That is, some events tend to co-occur while others are not allowed to co-occur at all. These constraints can be divided in 4 categories; from each category we will only mention the most salient, for further details see section 2 of Frank et al. (2009):

- Personal characteristics: Each person tends to play a particular game and with a particular toy. For example, Charlie likes and is skilled (tends to win) at playing chess. Likewise, Sophia likes to play soccer and Heidi likes to play hide&seek. Also, each person tends to be in some locations more often than others.
- Games and Toys: Each game and toy can be played only in certain locations. For instance, soccer can only be played in the street and a puzzle can only be played with in the bedroom. Some games/toys demand a specific number of participants, like chess that needs exactly 2 players. Each person can only be playing one game or with one toy at a time, and only a combination of playing soccer and playing with a ball is allowed.

- Being There: Each person can only be in one place at a time. If someone plays hide&seek in the playground, all players are in the playground; and the two players of chess are in the same place.
- Winning and Losing: People cannot win and lose at the same time, and there cannot be more than one winner. If someone wins, all other players lose, and if there is a loser, there must be one winner. Winning and losing can only happen while playing a game. Finally, winning is usually done "easily" by someone who plays "well" and "difficultly" by someone who plays "badly".

3.2.2 Situation Space

Having defined a microworld, the next step to obtain the desired representations corresponds to the construction of a *Situation Space*. Frank et al. (2009) do this by automatically generating a large number of 'observations' of the states-of-affairs in their microworld. A particular observation O_n is related to an instance in the microworld and is obtained by, stochastically and according to the probabilistic regularities of the microworld, setting each basic event to true (1) or false(0), yielding a binary vector $O_n \in [0,1]^K$, where K equals the number of basic events in the microworld (see Figure 3.2). The number of necessary observations N depends on the complexity of the interactions within the microworld. In general, it should be large enough, such that the sampled observations approximate the probabilistic nature of the microworld in terms of the (co-)occurrence probability of the basic events. Frank et al. (2009) sampled $25,000$ observations ($N = 25,000$) from their microworld. For the simulations presented here the same number of observations was used.[1]

After sampling the necessary observations, they are put together into a situation space matrix (see Table 3.2), with dimensionality $N \times K$, where N is the number of sampled observations and K is the number of basic events defining the microworld. Each column in the resulting matrix constitutes the *situation vector* of the basic event that is related to it. A situation vector encodes the meaning of an event in terms of the observations in which the event is true.

Following the notation of Frank et al. (2009), we refer to the situation vector related to an event a as $\mu(a)$, and $\mu(a) = (\mu_1(a), ..., \mu_n(a)) \in [0,1]^N$.

[1] Thanks to Harm Brouwer for providing the code to generate the situation space that we use here.

	play(charlie,chess)	play(charlie,hide&seek)	play(charlie,soccer)	...	manner(win,difficulty)
observation$_1$	1	0	0	...	1
observation$_2$	0	1	0	...	1
observation$_3$	1	1	0	...	0
...	.	.	.	...	.
observation$_N$	0	1	0	...	0

Table 3.2: Situation Space matrix.

3.2.3 Complex Events

A boolean combination of basic events is called a *complex event*. In propositional logic any boolean combination of propositions can be expressed using only the operators of negation and conjunction. Therefore it is only necessary to define these in order to be able to obtain the situation vector of any complex event.

Frank et al. (2009) defines these operations as is common in fuzzy logic:

$$\mu(\neg a) = 1 - \mu(a) \tag{3.1}$$

$$\mu_i(a \wedge b) = \mu_i(a)\mu_i(b) \tag{3.2}$$

With these two definitions the situation vector of any complex event can be computed. For example, a disjunction is defined as $a \vee b \equiv \neg(\neg a \wedge \neg b)$. Consequently, $\mu_i(a \vee b) = 1 - ((1 - \mu_i(a))(1 - \mu_i(b))) = \mu_i(a) + \mu_i(b) - \mu_i(a)\mu_i(b)$.

Situation vectors of basic and complex events correspond to the semantic representations sought by Frank et al. (2009). In general, they represent the meaning of an event in terms of the observations in which that event is true. This is similar to the approach of distributional semantics in which the meaning of a word is defined in terms of the words with which it appears, or the documents in which it appears.

3.2.4 Computing Probabilities

Situation vectors come with the benefit that they encode probabilistic information about their corresponding events. For a given event (either basic or complex), one can estimate the prior probability of that event by calculating the average over the components of the corresponding situation vector:

$$P(a) = \frac{1}{N} \sum_i \mu_i(a) \tag{3.3}$$

Similarly, the probability of a conjunction can be estimated by:

$$P(a \wedge b) = \frac{1}{N} \sum_i \mu_i(a)\mu_i(b) \tag{3.4}$$

with $a \neq b$. If $a = b$, then, Frank et al. (2009) defines $P(a \wedge a) = P(a)$.

The conditional probability of an event a given another event b ($P(a|b)$) is called the *belief value* of a given b, because it represents the extent to which a might be believed to be true, given b. This can be computed using the two previous equations:

$$P(a|b) = \frac{P(a \wedge b)}{P(b)} = \frac{\sum_i \mu_i(a)\mu_i(b)}{\sum_i \mu_i(b)} \tag{3.5}$$

According to the rules governing the microworld, not all complex events are possible or equally probable. For example, in the microworld defined by Frank et al. (2009), a person cannot both win and lose at the same time ($win(charlie) \wedge lose(charlie)$). Consequently, after performing the operations needed to obtain the corresponding situation vector, one would arrive to a vector filled with zeros, reflecting the probability of that event; likewise for all situation vectors related to complex events that are not allowed within the microworld.

It is important to note that these estimations are accurate to the extent that the number of sampled observations N is large enough to be representative of the microworld.

3.2.5 Dimensionality Reduction

As previously mentioned, the number of observations N has to be large enough in order to be able to obtain accurate probability estimates. However, if N is very large, the dimensionality of the situation space becomes impractically large. Thus, considering the $25,000$ observations sampled from their microworld, Frank et al. (2009) applied a dimensionality reduction technique in order to transform the situation vectors directly obtained from the observations into vectors with a more manageable dimensionality, while trying to maintain the probabilistic nature of the situation space. As Frank et al. (2009) explains, this procedure is merely a tool to obtain compressed representations, and is not intended to simulate the psychological process of developing event representations.

The dimensionality reduction technique consists of a self-organizing system called Competitive Layer, which contains M units. Each of these units is assigned K values, each one related to one of the K basic events defining the microworld. Putting these values together, one obtains a matrix similar to the situation space matrix, but with a dimensionality of $M \times K$, in contrast with the original situation space matrix with dimensionality $N \times K$.

During training, the units in the Competitive Layer adapt to the observations in an unsupervised manner resembling a Self-Organizing Map (Kohonen, 1995). The specific algorithm, as described in Frank et al. (2009), begins by associating each unit m to a weight vector $\mu_m \in [0,1]^{44}$ and a single bias unit b_m. Initially all weights are equal to 0.5 and all biases equal 1.

Training is performed during 20 epochs. During each epoch the following is performed for each observation O_n:

1. For each unit m, determine the cityblock distance between the weight vector of that unit and the current observation:

   $$d(\mu_m, O_n) = \sum_k |\mu_m^k - O_n^k| \tag{3.6}$$

   where μ_m^k and O_n^k refer to the value in the k-th dimension of μ_m and O_n respectively, which are related to the k-th basic event in the microworld.

2. Select the winner w, which is the unit with the shortest biased distance to O_n:

   $$w = argmin_m(d(\mu_m, O_n) - b_m) \tag{3.7}$$

3. Update the weight vector of the winner:

   $$\Delta\mu_w = \alpha(O_n - \mu_w) \tag{3.8}$$

   where α is the weight learning rate.

4. Decrease the bias of the winner (with a minimum of 1):

   $$\Delta b_w = \beta b_w(1 - b_w) \tag{3.9}$$

   where β is the bias learning rate.

S	$\longrightarrow$	$N_n VP_{n,v} APP_{n,v}$.
N_{person}	$\longrightarrow$	*charlie|heidi|sophia|someone|the boy|a girl*
N_{game}	$\longrightarrow$	*chess|hide&seek|soccer|football|a game*
N_{toy}	$\longrightarrow$	*a puzzle|a ball|a doll|a jigsaw|a toy*
$VP_{person,play}$	$\longrightarrow$	*plays*
$VP_{person,win}$	$\longrightarrow$	*wins|beats* N_{person}
$VP_{person,lose}$	$\longrightarrow$	*loses|loses to* N_{person}
$VP_{game,play}$	$\longrightarrow$	*is played*
$VP_{game,win}$	$\longrightarrow$	*is won*
$VP_{game,lose}$	$\longrightarrow$	*is lost*
$VP_{toy,play}$	$\longrightarrow$	*is played with*
$APP_{person,play}$	$\longrightarrow$	$[N_{game}][Manner][Place]\vert PP_{toy}\ [Place]\vert Place\ PP_{toy}$
$APP_{person,win}$	$\longrightarrow$	$[PP_{manner}][PP_{game}][Place]\vert PP_{game}\ PP_{manner}\vert$ $Place\ PP_{game}$
$APP_{person,lose}$	$\longrightarrow$	$[PP_{game}][Place]\vert Place\ PP_{game}$
$APP_{game,play}$	$\longrightarrow$	$[Manner][Person][Place]$
$APP_{game,win}$	$\longrightarrow$	$[Manner][Person][Place]$
$APP_{game,lose}$	$\longrightarrow$	$[PP_{person}][Place]$
$APP_{toy,play}$	$\longrightarrow$	$[PP_{person}][Place]\vert Place\ PP_{person}$
$Manner$	$\longrightarrow$	*well|badly*
$Place$	$\longrightarrow$	*inside|outside|*PP_{place}
PP_{place}	$\longrightarrow$	*in the bathroom|in the shower|in the bedroom|in the street| in the playground*
PP_{person}	$\longrightarrow$	*by* N_{person}
PP_{game}	$\longrightarrow$	*at* N_{game}
PP_{toy}	$\longrightarrow$	*with* N_{toy}
PP_{manner}	$\longrightarrow$	$with\ ease\vert with\ difficulty$

Table 3.3: Grammar of the Microlanguage. Variable $n \in \{person, game, toy\}$ denotes nouns; $v \in \{play, win, lose\}$ denotes verbs; VP=verb phrase; APP=adverbial/prepositional phrase; PP=prepositional phrase. Items in square brackets are optional.

5. Increase the bias of all losers, with $m \neq w$:

$$\Delta b_m = \beta b_m \tag{3.10}$$

α and β were initially set to 1 and 0.0001 respectively. After each of the first 10 training epochs, their values were reduced linearly to end up at 10% of the initial values. During the last 10 epochs, they remain at these levels.

In Frank et al. (2009), and as previously stated, $N = 25,000$ and $K = 44$. Additionally, the value of M was chosen according to the quality of the resulting vectors, which was measured by comparing the true (conditional) probabilities in the microworld to the corresponding belief values obtained from the reduced situation vectors. Larger values of M generally lead to greater correlation coefficients, implying better results. Having $M = 150$, the correlation coefficient was very good ($r \geq 0.996$), with little improvement for larger values of M. Therefore Frank et al. (2009) set it to 150.

This process allows us to obtain situation vectors for each basic event with a dimensionality $M = 150$. Situation vectors for complex events can be obtained through conjunction and negation operations, as previously described, in this case using the vectors with reduced dimensionality.

3.2.6 Frank et al. (2009)'s Microlanguage and Examples Set

Events in the microworld are described by sentences obtained from a *microlanguage*. This microlanguage consists of 40 words that can be combined into 13556 sentences according to its grammar. For the simulations in this thesis, the grammar of Frank et al. (2009) was minimally modified by introducing the determiners "a" and "the", and an end-of-sentence marker, leaving a total of 43 words in the vocabulary. The resulting grammar can be seen in Table 3.3.

During the process of sentence generation, propositional logic semantics are attached to each sentence (see examples shown in Table 3.4). These in turn are later converted to situation vectors by applying the operations described above (in order to obtain situation vectors for complex events) over the situation vectors related to the basic events contained in the propositional logic semantics.

Using the grammar of the microlanguage, a set of sentences with their corresponding situation vectors was generated.[2] This set comprises the dataset on which the model of Frank et al. (2009) was trained, as well as the models presented here.

The grammar allows the generation of 13556 sentences, however, many of the sentences describe situations that are not allowed according the the rules governing the microworld, consequently having empty situation vectors. These were removed, leaving a total of 8201 sentences.[3] From these, 6782 sentences were in active voice and 1419 in passive. Note that this set contains sentences with simple semantics (e.g., "charlie plays chess." → $play(charlie, chess)$), as well as sentences with complex semantics (e.g., "a girl plays chess." → $play(heidi, chess) \vee play(sophia, chess)$).

As one can see in the first two examples in Table 3.4, many sentences are related to the same semantic representations. In total, there are 782 unique semantic representations, from which 424 are related to both passive and active sentences. The rest (358)

[2] Thanks to Harm Brouwer for providing the code to generate the sentences with their corresponding situation vectors.

[3] A sample of the sentences can be found in Appendix A, a full list can be found in: `https://github.com/iesus/thesis-production-models/blob/master/sentences.txt`

Sentence	**Semantics**
charlie plays chess.	$play(charlie, chess)$
chess is played by charlie.	$play(charlie, chess)$
sophia plays with a ball in the street.	$play(sophia, ball) \wedge place(sophia, street)$
someone plays with a doll.	$play(charlie, doll) \vee play(sophia, doll) \vee play(heidi, doll)$
charlie loses to sophia.	$win(sophia) \wedge lose(charlie)$
sophia beats charlie at chess.	$win(sophia) \wedge lose(charlie) \wedge play(sophia, chess)$
charlie wins inside.	$win(charlie) \wedge (place(charlie, bedroom) \vee place(charlie, bathroom))$
sophia plays soccer well.	$play(sophia, soccer) \wedge manner(play(sophia), well)$

Table 3.4: Examples of sentences generated with the microlanguage, paired with the propositional form of the described event.

corresponds to situations that can only be expressed by active sentences according to the grammar. More concretely, the grammar does not define passive sentences for situations where the object of the action is either a person (e.g. "Heidi beats Charlie.") or undefined (e.g. "Charlie plays."). While the grammar of Frank et al. (2009) could be extended in order to define passive constructions for these situations, it was left as it is in order to inspect the behavior of the models with regards to generalization.

Putting together sentences related to the same semantics, and separating active and passive sentences, a dataset was constructed. The dataset consists of a set of pairs $\{(DSS_1, \varphi_1), \ldots, (DSS_n, \varphi_n)\}$ where each DSS_i corresponds to a situation vector plus a bit indicating if the pair is related to active (1) or passive (0) sentences ($DSS_i \in [0,1]^{151}$); and $\varphi_i = \{sent_1, \ldots, sent_k\}$ where $sent_j$ is a sentence, a sequence of words $word_1, \ldots, word_n$, expressing the information contained in DSS_i. Each set φ_i represents all the possible sentences that express the information contained in the corresponding DSS_i and in the expected voice. In total, there are 1,206 (DSS_i, φ_i) pairs.

3.3 Belief Vectors

The unreduced situation vectors introduced by Frank et al. (2009) explicitly mark in which observations the given event is true, out of the $25,000$ sampled observations. For each observation, if the given event is true, there will be a 1 in that dimension, 0 otherwise. After applying the dimensionality reduction, the 25,000 observations are reduced to a set of 150 pseudo-observations, in which the probabilistic structure of the microworld is loosely preserved. The resulting 150-dimensional situation vectors are more manageable and can serve as input or output to a computational model, which is their main purpose.

Nonetheless, as consequence of the dimensionality reduction, some information is lost during the process. Concerning the microworld defined by Frank et al. (2009), information regarding adverbial modification, such as "well", "badly", "with ease" and "with difficulty", is no longer available. For the sake of comparison, the 150-dimensional situation vectors defined by Frank et al. (2009) were used first for the simulations presented in the next chapters, but later on the focus was given to *belief vectors*. These vectors are derived from the original $44 \times 25,000$ dimensional matrix and could be regarded as an alternative way to obtain vectors with reduced dimensionality.

In general, a belief vector corresponds to the most likely state-of-affairs of the microworld, given a (complex) event. This is calculated by taking all the observations (amongst the 25,000 sampled) in which the event is true, and then averaging their states-of-affairs.

Basic Event i	**P(i|sent)**
$play(charlie, chess)$	1.0
$play(charlie, hide\_and\_seek)$	0.0
$play(charlie, soccer)$	0.0
$play(heidi, chess)$	0.51138406
$play(heidi, hide\_and\_seek)$	0.0
$play(heidi, soccer)$	0.0
$play(sophia, chess)$	0.48861594
$play(sophia, hide\_and\_seek)$	0.0
$play(sophia, soccer)$	0.0
$play(charlie, puzzle)$	0.0
$play(charlie, ball)$	0.0
$play(charlie, doll)$	0.0
$play(heidi, puzzle)$	0.07789095
$play(heidi, ball)$	0.10515279
$play(heidi, doll)$	0.15248652
$play(sophia, puzzle)$	0.0724985
$play(sophia, ball)$	0.17765129
$play(sophia, doll)$	0.09916117
$win(charlie)$	0.29448772
$win(heidi)$	0.07789095
$win(sophia)$	0.07759137
$lose(charlie)$	0.15548232
$lose(heidi)$	0.15368484
$lose(sophia)$	0.14080288
$place(charlie, bathroom)$	0.0
$place(charlie, bedroom)$	1.0
$place(charlie, playground)$	0.0
$place(charlie, street)$	0.0
$place(heidi, bathroom)$	0.06590773
$place(heidi, bedroom)$	0.68544038
$place(heidi, playground)$	0.14529658
$place(heidi, street)$	0.1033553
$place(sophia, bathroom)$	0.07579389
$place(sophia, bedroom)$	0.63780707
$place(sophia, playground)$	0.16536848
$place(sophia, street)$	0.12103056
$manner(play(charlie), well)$	0.1758538
$manner(play(charlie), badly)$	0.20581186
$manner(play(heidi), well)$	0.09556621
$manner(play(heidi), badly)$	0.09946075
$manner(play(sophia), well)$	0.09496705
$manner(play(sophia), badly)$	0.09137208
$manner(win, easily)$	0.07219892
$manner(win, difficultly)$	0.05152786

Table 3.5: Belief vector related to the sentence “charlie plays chess.”.

This is not necessarily in conflict with a neural interpretation of the original situation vectors. With the original 25,000 dimensional situation vectors, the meaning of an event is represented by activating all observations in which the event is true. Thus, assuming that each observation itself has an internal representation and that similar representations would share common structural elements, it is not difficult to imagine that the activation of all observations in which one event is true would be similar to the activation of the mean state-of-affairs across all observations in which the event is true. For example, let us imagine 3 observations that are related to a sentence. These observations have 2 features in common but differ in a third one. If we assume that each feature is represented neurologically only once, then activating the 3 observations at the same time (thus activating the meaning of the sentence) would imply the activation of the 2 common features and each of the individual features. If activation is cumulative, then the activation pattern would be a vector where the 2 common features would be activated 3 times and each of the individual features only once. This is equivalent to a belief vector, except that the belief vector would have an extra step corresponding to the normalization in order to transform the sum over the activation values into probabilities.

An alternative and equivalent way to obtain belief vectors is by taking the dot product between the original $44 \times 25,000$ dimensional situation space matrix and the 25,000 dimensional situation vector that is associated to the complex event, and then dividing each dimension of the resulting vector by the number of ones in the original 25,000 dimensional situation vector:

$$bv_i = \frac{A \cdot sv_i}{\sum\limits_j sv_i^j} \tag{3.11}$$

where A is the $44 \times 25,000$ dimensional situation space space matrix, sv_i is the 25,000 dimensional situation vector of the event i, and j is an index related to each dimension in sv_i. The resulting vector bv_i contains 44 dimensions, each one associated to each basic event in the microworld, and the value of each dimension is equal to the conditional probability of each basic event given the complex event.

As an example, Table 3.5 shows the belief vector related to the sentence "Charlie plays chess.". One can see that the vector reflects the implications of the sentence: first, the dimension related to $play(charlie, chess)$ is set to 1, while all other dimensions of the form $play(charlie, \_)$ are set to 0. Then, $place(charlie, bedroom)$ is also set to 1 because chess can only be played in the bedroom. Since chess has to be played by 2 people, and there are 3 people in the microworld, the sentence implies that either Heidi or Sophia is playing with Charlie, and as shown by the dimensions $play(heidi, chess)$ and $play(sophia, chess)$, Heidi is slightly more likely to be playing with Charlie. The rest of the dimensions follow a similar pattern giving an intuition of the state-af-affairs of the microworld, given the sentence.

Intuitively, the meaning of a sentence should reflect all observations that are in accordance with the sentence, without enumerating each and all of the possible observations, as this number might not be finite. Moreover, people presumably do not store each detail of each observation they experience in memory, rather they likely store abstractions, which allows them to relate similar observations even if they are not exactly the same. The proposed belief vectors would account for that, offering an abstraction over the set of observations related to a sentence, namely their average.

One additional benefit of belief vectors is that their dimensionality is lower and is given by the number of basic events in the microworld, and it is not another hyperparameter.

Finally, belief vectors are human interpretable, as each dimension is related to a basic event. This is an aspect that later will become useful in Chapter 6, where an analysis of the internal mechanism of the model is given, and for which human interpretable input representations are important.

Using equation 3.11 and the original 25,000-dimensional situation vectors, a dataset was constructed, similar to the one described in the previous section, with the only difference being that the 150-dimensional situation vectors are replaced by the 44-dimensional belief vectors. For the simulations presented in the next chapters, this dataset was used as training/testing set, together with the set of 150-dimensional situation vectors.

3.4 Situation and Belief Vectors

In the previous sections the process to obtain the semantic representations (situation and belief vectors) used in this thesis was shown. In this section, some of their main properties will be explained and elaborated, showing their commonalities and the advantages/disadvantages of each type of representation.

3.4.1 Probabilities

The main feature sought by Frank et al. (2009) was direct inference. Their situation vectors possess this feature, since for any pair of events A and B, one can calculate P(A) and P(A|B) using only their situation vectors, allowing the comparison of any pair of events, drawing relations between them just by looking at their representations.

Being able to assess similarity between entities by looking at their representations is fundamental if one expects a computational model to be able to generalize to new inputs/outputs. In the DSS model, each event is represented by a point in a multidimensional space, where similarity can be quantified by measuring the distance between vectors. Consequently, any point within this space can be related to other points, even if it has not been seen before. In contrast, using localist representations, each representation is independent of each other having different sets of connection weights, so any relation or similarity between representations has to be learned during training.

Both situation and belief vectors are points in a multidimensional space. Additionally, in both cases one can measure the similarity between representations by measuring the distance between vectors. Nonetheless, belief vectors are formed by conditional probability values, where each dimension of a belief vector related to an event A has a value of $P(B_i|A)$, where B_i corresponds to a basic event. As a consequence, prior probabilities

$(P(A))$ cannot be obtained from these representations. Likewise, conditional probability values $(P(B|A))$ are only available for basic events. That is, one can only know the state of the microworld given a sentence in terms of the basic events, but cannot infer other conditional probabilities. For example, having the belief vector of the sentence "charlie plays chess.", one can know the probability of Charlie winning $(win(charlie))$, but one cannot know the probability of Sophia playing with a puzzle in the bedroom $(play(sophia, puzzle) \wedge place(sophia, bedroom))$ given that Charlie plays chess.

The ability to calculate prior and conditional probabilities from the representations might not always be helpful, specially for events with probabilities approaching zero. Considering a more realistic (micro)world with more basic and therefore more complex events, the prior probability of a large proportion of the complex events would be close to zero, reflecting the difficulty of predicting events when the number of factors governing a (micro)world increases. Furthermore, normal language use includes the expression of events that are imaginary, contradictory, and other instances where an event might never be observed. If the probability of a complex event approaches or is equal to zero, the corresponding situation vector would as well approach a vector filled with zeros, giving little insight about the related sentence. In such cases, belief vectors can be constructed by setting each dimension to a hypothetical truth value even if the microworld would not otherwise permit it. Thus, while they do not allow the computation of all the probabilistic information that situation vectors contain, they can still be used as a representation of events with probability equal or close to zero.

In sum and concerning probabilistic information, on the one hand, if the computation of prior and conditional probabilities is important, then situation vectors would be the more suitable representation to use. On the other hand, if the representation of rare or impossible events (according to the microworld) is important, then belief vectors would be a better option. In both cases, the representations are points in a multidimensional space, where one can compute similarity values by measuring the distance between vectors.

3.4.2 Information Loss

As previously explained, the dimensionality reduction technique used to obtain the 150-dimensional situation vectors implies information loss of aspects regarding adverbial modification. In the simulations presented in the next chapters, this loss implied that in many cases the model was unable to differentiate some events, which motivated the introduction of belief vectors.

Nevertheless, going from the original 25,000-dimensional situation vectors to the 44-dimensional belief vectors introduces a different kind of information loss. If two sentences are equivalent, their situation vectors would point to the same observations, and therefore, their mean state-of-affairs would be equal. Then, equivalent sentences would have the same belief vectors and situation vectors, as expected. However, it is possible that different sentences related to different observations would have the same mean state-of-affairs and thus belief vectors. This would be possible for pairs of highly ambiguous sentences; but even then, considering that it is a multidimensional continuous space, the probability of two vector representations coinciding in the exact point is very low.

One can also imagine a sentence portraying two different clusters of observations. For example, assuming the hypothetical sentence “the girl sees the man with the telescope”, this sentence would refer to two different clusters of observations: one where the prepositional phrase “with the telescope” is attached to the verb “sees”, and another one where the prepositional phrase is attached to “the man”. Then, the average over all the observations, or belief vector, would be a point in the middle, with features related to both clusters. While the resulting vector might still be uniquely related to the sentence, the vector would be unable to represent the details of two situation models (one per cluster of observations) at the same time, and therefore, we could say that information regarding the details of each situation model would be partially lost.

It is unclear how many situation models people are able to construct simultaneously and maintain in mind. While incremental comprehension implies discrimination over possibly a large number of situations, the resulting situation model would be mostly disambiguated by the end of the sentence, possibly rendering a single cluster of observations. Furthermore, for the case of language production, we can assume that people would maintain a single situation model. This is similar to the case of polysemous words, where a single word can refer to a set of possibly very different concepts; however, during production speakers would normally refer to a single one.

For the set of sentences and meaning representations that are used for the simulations in the next chapters, each sentence maps to a single cluster of observations uniquely identified by its mean state-of-affairs, such that equivalent sentences map to the same belief vectors, and also if two sentences map to the same belief vector, it implies that they are equivalent.

In sum, 150-dimensional situation vectors preserve to some extent the information of the 25,000-dimensional original situation vectors, by enumerating amongst the 150 pseudo-observations, the ones that are in accordance with a given event; however, some aspects of the representations are lost, which prevents the models from differentiating some events.

In contrast, using belief vectors one can differentiate each semantic representation in an exact manner; however, the specific details of each observation related to a particular event are also lost, remaining only the average over the relevant observations.

It is worth notice that in case one wants to use situation vectors, recently Venhuizen et al. (2018) used a different dimensionality reduction technique with the potential of avoiding the information loss described previously.

3.4.3 Non-symbolic analogous representations

Another motivation for the situation vectors of Frank et al. (2009) was that symbolic localist representations do not contain information in the representation themselves about the nature of what is being represented, and therefore one cannot make comparisons between the represented items. This aspect of localist representations would hinder the systematicity of a computational model, which as mentioned before, is an important property to be considered during computational modeling of human sentence processing. In exchange of localist representations, Frank et al. (2009) proposed analogous representations.

An analogous representation is one where the form of the representation depends on what is being represented, reflecting its properties, and it is not an arbitrary assignment. For example, the word "chair" is a symbolic representation of the concept CHAIR, since one would need to know English in order to have an idea of its meaning. In contrast, a picture of a chair would be an analogous representation, since one can immediately grasp the nature of what the representation refers to, as it is embedded in the form of the representation. Additionally, one can immediately assess similarity between a chair and a table by looking at their pictures, while one cannot say much about their similarity by looking at the words "chair" and "table" without knowing English.

A representation is analogous to the extent that it contains the information that defines a given entity and that permits us to differentiate that entity from others. Symbolic representations do not contain much information apart from the identity of what is being represented. For example, using localist representations, entities are represented by a vector where only one dimension is set to 1, while the rest are set to 0. This vector does not reflect any further information of what is being represented, and therefore, such information would have to be learned during training. This means that any novel representation would be unable to be processed without an amount training.

Situation and belief vectors are analogous representations that not only represent the identity of their corresponding events, but also they include very rich information about the specific state-of-affairs of the microworld related to those events, thereby allowing for ‘world knowledge’-driven inference. Consequently, events can be compared to other events, establishing relations of similarity and/or implication, just by looking at their representations.

Chapter 4

Sentence Production Model

This chapter presents the computational model of sentence production that is proposed in this thesis and that was used for the simulations presented in the next chapters.

Sentence production involves the mapping of a message representation to a sequence of words forming a sentence. In this process, not only must the appropriate words be selected, but also they have to be retrieved in the correct order, such that the sentence conveys the intended message representation, while meeting the grammatical constraints of the language. Consequently, this process corresponds to Grammatical Encoding, according to Levelt (1989)'s model (see Chapter 2).

According to the model of Levelt (1989), during grammatical encoding a message representation is taken as input. Then, the lemmas matching parts of the message are activated, which in turn activate syntactic information. Using this information, syntactic structures are built, such as noun or verb phrases, which put together form a sequence of lemmas, corresponding to a sentence.

In the model presented here, the situation/belief vectors described in Chapter 3 are taken as message representations in order to produce sentences that describe the associated events. The output of the model corresponds to a sequence of word representations that do not contain phonological information. As such, the scope of this model concerns only grammatical encoding, being independent, on the one hand, of the processes that might construct the message representations; and on the other hand, of processes concerning phonological encoding.

Encoding at a sentence level is a much more productive process than lexical access or phonological encoding, which mostly involve the retrieval of elements from memory (Dell et al., 1999). Indeed, apart from instances where productive morphology can produce new words, lexical access is restricted to the retrieval of items from a finite mental lexicon. In contrast, the number of sentences that can be generated using the grammar of a language and its lexicon is infinite. Consequently, an important aspect of a model of sentence comprehension/production is that it must be able to handle sentences that have not seen during training, showing systematicity (see Fodor and Pylyshyn, 1988).

Other aspects that need to be considered are that human language production is incremental, where the production of each word depends on the previous ones, showing an overall coherence that follows the syntactic rules of the language; and that the statistical patterns of the language are reflected in the behavior of the speakers (e.g., Bell et al., 2003; Hale, 2001b; Jurafsky, 1996). In the rest of this chapter, the architecture of the model and its training procedure are explained, showing how the model approaches incrementality and how it learns the statistical patterns of the language.

The main code that implements this model can be found in appendices B and C. A full version can be found in `https://github.com/iesus/thesis-production-models`.

4.1 Architecture

The model architecture, shown in Figure 4.1, consists of an input layer containing the representation of the message to be conveyed, a 120-unit recurrent hidden (sigmoid) layer, and a 43 unit (softmax) output layer where each unit corresponds to a word in the vocabulary.

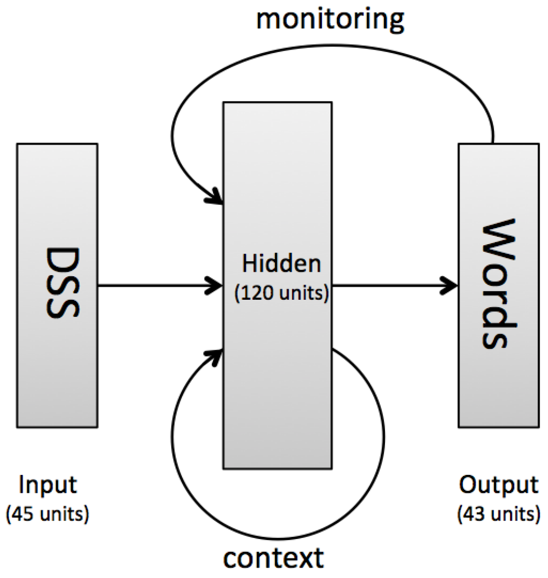

FIGURE 4.1: Model architecture.

The dimensionality of the input layer is determined by the chosen semantic representation (150-dimensional situation vector or 44-dimensional belief vector), plus one bit indicating if the model should produce an active sentence (1) or a passive one (0). Thus, the input layer dimensionality is equal to 151 when using the original 150-dimensional situation vectors of Frank et al. (2009), and 45 when using belief vectors. The dimensionality of the output layer matches the number of available words in the vocabulary plus the end-of-sentence marker (43).

Time in the model is discrete. At each time step t, activation of the input layer dss is propagated to the hidden recurrent layer. This layer also receives a copy of its own activation h_{t-1} at time-step $t-1$ (zeros at $t=0$) through *context units.* These units help to preserve some memory of what has been produced expanding several time steps in the past. Additionally, the hidden layer receives the identity of the word mon_{t-1} produced at time-step $t-1$ (zeros at $t=0$) through *monitoring units*, where only the unit corresponding to the word produced at time-step $t-1$ is activated (set to 1).

More formally, activation of the hidden layer is given by:

$$h_t = \sigma(W_{ih} \cdot dss + W_{hh} \cdot h_{t-1} + W_{mh} \cdot mon_{t-1} + b_h) \tag{4.1}$$

where W_{ih} is the weight matrix connecting the input layer to the hidden layer, W_{hh} is the weight matrix connecting the hidden layer to itself, W_{mh} is the weight matrix connecting the monitoring units to the hidden layer, and b_h corresponds to the bias unit of the hidden layer.

Then, the activation of the hidden layer h_t is propagated to the output layer, which yields a probability distribution over the vocabulary, and its activation is given by:

$$output_t = softmax(W_{ho} \cdot h_t + b_o) \tag{4.2}$$

where W_{ho} is the weight matrix connecting the hidden layer to the output layer and b_o is the vector corresponding to the output bias unit.

The word produced at time-step t is defined as the one with highest probability (highest activation). The model stops when an end-of-sentence marker (a period) has been produced.

As an example, Figure 4.2 illustrates the production of the sentence “someone plays badly.”. At time step t=0, the DSS representation is fed into the hidden recurrent layer, which in turn propagates its activation to the output layer, the output layer then yields a probability distribution over the vocabulary. In this case, the words “someone” and “a” have high activation. Given that “someone” has the highest activation, it is the word that the model produces. At time t=1, the hidden recurrent layer is fed again with the DSS representation, but this time also the activation of the recurrent hidden layer at t=0 and the identity of the word produced also at t=0 (“someone”). It then propagates its activation to the output layer, which again yields a probability distribution over the vocabulary. This time the only activated word is “plays”. At time t=2, the hidden layer again receives the activation of the DSS representation, the activation of the hidden layer at t=1, and the identity of the word produced at t=1 (“plays”). It then propagates its activation to the output layer, which in this case activates the words “badly” and “a”. Since ”badly” has higher activation than “a”, it produces “badly”. Finally, the process is repeated at t=3, but this time the model produces “.”, which signals the end of production.

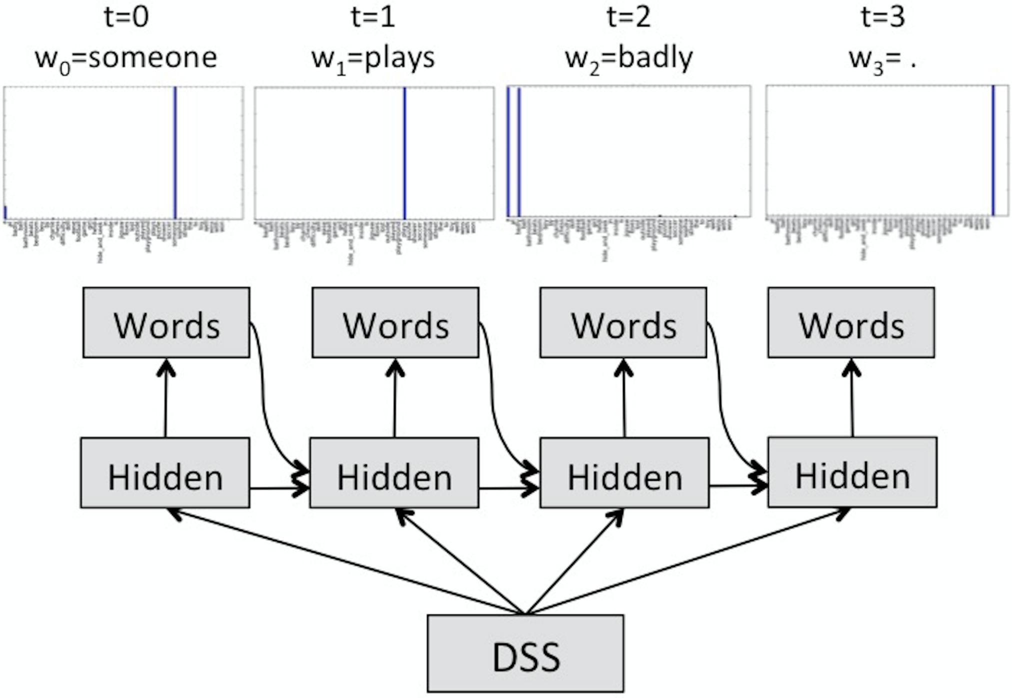

FIGURE 4.2: Example of Production:“someone plays badly.”

The recurrent nature of this architecture, permits the model to behave incrementally, producing one word per time step. As a result, a sequence of words produced in this way presents coherence, as each word depends, on the one hand, on the message representation, and on the other hand, on the words that have been previously produced, allowing the network to enforce grammatical constraints. Additionally, in contrast to a feedforward architecture, a recurrent architecture permits the generation of sentences of variable length, since the number of words produced is not given beforehand, but is chosen by the model itself by producing an end-of-sentence marker.

4.1.1 Similar Architectures

The architecture proposed was inspired by the comprehension model of Frank et al. (2009), with the main difference being that the inputs and outputs are reversed, and the inclusion of monitoring connections. As Frank et al. (2009) explain, this is a very simple architecture, which facilitates the assessment of the importance of the input/output representations, with respect to the complexity of the architecture.

Indeed, as it is a production model, the architecture is even more similar to the Phonological Error Model (see Chapter 2, Dell et al., 1993), with the main difference being that the Phonological Error Model generates sequences of phonemes given a lexical representation, while the model proposed here generates sequences of words given a message representation. In both cases, the input is a fixed representation that produces a sequence of tokens of variable length. And similar to Dell et al. (1993), this architecture is able to draw dependence relations between the tokens produced, which in their case allowed them to account for phonological accommodation and other phonological phenomena.

Compared to the architecture of the Structural Priming Model (see Chapter 2, Chang et al., 1997), the architecture presented here is also similar, with the main difference being that the context units are not linked to comprehension as in Chang et al. (1997), where the content of these units was set manually and trying to approximate the output of comprehension. In the architecture presented here, the context units are not meant to reflect comprehension processes, but rather they are used as memory units that preserve a history of what has been produced so far. Also, their content is not set manually but it is determined by the activation of the hidden layer at time step $t - 1$.

The model presented here is perhaps most similar to the Prod-SRN model (see Chapter 2, Chang, 2002). However, the semantic representations used by Prod-SRN, as well as the production models previously mentioned, are significantly different. In their case, they use localist binding-by-space representations, while the model presented here uses the DSS representations described in Chapter 3, which are continuous and distributed representations. While Chang (2002) tested the Prod-SRN model for systematicity, exhibiting less systematicity than the Dual-Path model, it is unclear to what degree that was due to the message representations, rather than to the architecture.

Compared to the Dual-Path model (see Chapter 2, Chang, 2002), the model presented here shares the use of a recurrent layer and the general flow of activation is to some degree similar. The main difference is that the Dual-Path model introduces a binding-by-weight mechanism merging the message representation with the architecture, and that the computation is split into 2 different paths, resulting in a much more complex architecture. While such architecture was successful in showing systematicity, the message representations had to be embedded into the architecture, which meant that for each new representation, the wiring of some parts of the model had to be changed. It is not clear how such a mechanism would be implemented neurally, and furthermore, whether such a mechanism is necessary if one uses more expressive message representations.

Finally, one can also recognize a similarity of the model proposed here with some architectures used in computational linguistics, most notably in language modeling and machine translation (e.g., Mikolov et al., 2010; Sutskever et al., 2014). Such models show high performance in their related tasks, being able to generalize to new input representations in an open domain context. In those cases, localist representations of words are first converted to word embeddings, where each word is a point in a multidimensional continuous space, similar to the DSS representations presented in the previous chapter.

4.2 Training

The process of training consists of showing the model examples of how it is supposed to behave for a given input. Each time a training example is given, the model changes its connection weights in order to approach the expected output. After several repetitions of the training examples, and if the training process is successful, the model would have learned to exhibit the expected behavior.

Since this process relies on experience, models trained in this manner usually are able to learn the patterns in the training examples according to their statistical properties. For the production model presented here, the model is given examples of how to map a semantic representation onto a sentence. Through repetition of these examples, the model is able to identify what patterns are statistically permitted in the language, in the semantic representations, and in the mapping between them. All this knowledge is obtained implicitly, as the model is not given explicit syntactic or semantic rules, only pairs of semantics and sentences.

One could relate this process of learning through experience to the process of human language development, which is an approach that indeed some models pursue (e.g., Chang et al., 2006). However, folowing Frank et al. (2009), this work is not intended to model language development, and therefore, we make no claims in that respect.

4.2.1 Training Set

A training set for this model consists of pairs of the form $(DSS, sent)$, where DSS is a semantic representation, either a situation vector or a belief vector depending on the chosen type of semantic representation for a given simulation, and *sent* is one of the sentences that conveys the message of DSS.

More precisely, in order to generate a training set, we begin by taking the dataset described in the previous chapter, which consists of a set $\{(DSS_1, \varphi_1), \ldots, (DSS_n, \varphi_n))\}$ where each DSS_i corresponds to a situation or belief vector, plus an additional unit used to indicate if the pair is related to active (1) or passive (0) sentences; and $\varphi_i = \{sent_1, \ldots, sent_k\}$ where $sent_j$ is a sentence expressing the information contained in DSS_i.

Then, considering a particular DSS_i, for each sentence $sent_j$ in φ_i, create a pair $(DSS_i, sent_{i,j})$, where $sent_{i,j}$ is the j-th sentence related to the i-th DSS, and add it to the training set. Doing this over all pairs in the original dataset generates a set:

$$\{(DSS_1, sent_{1,1}), \ldots, (DSS_1, sent_{1,k}), (DSS_2, sent_{2,1}), \ldots, (DSS_n, sent_{n,k})\}$$

where k corresponds to the total number of sentences related to each DSS_i, which varies according to each DSS_i. The resulting number of pairs obtained through this process is equal to the number of sentences generated by the microlanguage that are allowed according to the rules in the microworld. For the microlanguage of Frank et al. (2009), this number is 8,201.

4.2.2 Training Procedure

Prior to training, all weights on the projections between layers are initialized with random values drawn from a normal distribution $\mathcal{N}(0, 0.1)$. The weights on the bias projections are initially set to zero. Then the model is trained using cross-entropy backpropagation Rumelhart et al. (1986), where at each time step the model is expected to produce the word of the training sentence that corresponds to that time step. Weights are updated accordingly after each word in the sentence of each pair $(DSS, sentence)$ in the training set.

During training, the monitoring units are set at time t to what the model was supposed to produce at time $t - 1$ (zeros for $t = 0$). This reflects the notion that during training the word contained in the training sentence at time-step $t - 1$ should be the one informing the next time step, regardless of the previously produced (and possibly different) word. During testing, the monitoring units are set to 1.0 for the word that was actually produced and 0.0 everywhere else.

Unless otherwise specified, for the simulations in the next chapters the model is trained for a maximum of 200 epochs, each epoch consisting of a full presentation of the training set, which is randomized before each epoch.

An initial learning rate of 0.124 is employed, which is then halved each time there is no improvement of performance on the training set during 15 epochs. No momentum is used. Finally, training halts if the maximum number of epochs is reached or if there is no performance improvement on the training set over a 40-epoch interval.

4.2.3 Testing

After training, one can test the model to verify if a specific behavior was obtained. The nature of the procedure would depend on the aspect that one wants to test. For example, a common practice in computational linguistics, and the machine learning community in general, is to test the models to see if they are able to generalize to instances that do not belong to the training set. In that case, a set of input-output pairs are separated from the set of training examples, such that the model never sees those examples during training. Then, the model is tested in order to see whether it is able to correctly process those excluded items, in which case one can say that the model is able to generalize. The degree of generalization would depend on how different the training items are from the testing items.

Following these notions, the next chapter evaluates the model in terms of generalization or systematicity, where the model is tested to see whether it is able to produce novel sentences for novel semantic representations. In addition we analyze the production patterns that are exhibited in those novel sentences. Finally, in Chapter 7, an extension of this model is tested in order to see the behavior of the model while producing sentences according to different policies, and how they approach the UID Hypothesis. The details of the testing procedures are given in each of those chapters.

Chapter 5

Connectionist Systematicity

Language production and comprehension involve the mapping between sentences and message representations. As mentioned in previous chapters, an important challenge in such mapping is that both the number of sentences of a language and the number of the related message representations is infinite. As a consequence, it is impossible to memorize or even enumerate all possible sentences with their related message representations. Then, a model of language production or comprehension necessarily needs to be able to process novel sentences and message representations, showing systematicity.

Systematicity refers to the ability to generalize from a set of known instances of a particular phenomenon to a set of novel instances, by profiting from the regularities between the known and the novel ones. In terms of human language processing: "the ability to produce/understand some sentences is intrinsically connected to the ability to produce/understand certain others" (Fodor and Pylyshyn, 1988, p. 37). This property has been proposed to be ubiquitous in human cognition, and even a law of cognitive systems Fodor and McLaughlin (1990); Fodor and Pylyshyn (1988).

Concerning systematicity, Fodor and Pylyshyn (1988) originated a debate arguing that connectionist cognitive models are unable to behave systematically, and even if they did, they would need to implement a symbolic system, similar to the one proposed by the Language of Thought Hypothesis (Fodor, 1975), where the cognitive system consists of a set of rules operating over a set of symbols with combinatorial dynamics and internal hierarchical structure. If that is the case, connectionist models are reduced to descriptions at the implementational level of analysis, with little to no explanatory value at the algorithmic level.

Since the beginning of this debate, proponents of connectionism have argued that connectionist models are able to exhibit systematicity (for a review see Symons and Calvo

(2014)), from a theoretical point of view (e.g., Bechtel, 1993; Gelder, 1990) and empirically (e.g., Bodén, 2004; Brakel and Frank, 2009; Chang, 2002; Elman, 1991). However, until recently some issues of the debate still remain open. Among these, one can mention:

a) whether connectionist models can behave systematically,

b) whether they would be implementing a symbol system,

c) the reason for systematicity, that is, under which circumstances systematic behavior is expected as an implication of the architecture and not just as a mere coincidence.

Although nowadays it is fairly accepted that connectionist models can show systematicity to some degree, it has been argued that they need not only show the existence of systematicity, but also they need to show a level of systematicity comparable to a human level.

In order to operationalize and measure systematicity, Hadley (1994a) proposed to define it in terms of learning and generalization, where a neural network is said to behave systematically if it is able to process inputs for which it was not trained. Then, the level of systematicity would depend on how different the training items are from the testing items. Along this line, Hadley (1994a,b) proposed that human level systematicity is achieved if the model is able to behave with semantic systematicity, which is the ability to construct correct meaning representations of novel sentences.

In this context, many language comprehension models have been proposed (e.g., Farkaš and Crocker, 2008; Hadley and Cardei, 1999; Hadley and Hayward, 1997; Jansen and Watter, 2012; Miikkulainen, 1996). Of particular relevance for our matters is the approach of Frank et al. (2009), which develops a connectionist model of comprehension which is argued to achieve semantic systematicity. Their sentence comprehension model takes a sentence in the form of a sequence of words, and constructs a situation vector. As described in Chapter 3, each situation vector corresponds to a *situation model* (see Zwaan and Radvansky, 1998) of the state-of-affairs described by a sentence that also incorporates "world knowledge"-driven inferences. When the model processes a sentence like 'a boy plays soccer', for instance, it not only recovers the explicit, literal propositional content, but also constructs a more complete situation model in which a boy is likely playing outside on a field, with a ball, with others, and so forth. In this way it differs from other connectionist models of language comprehension and production, that typically employ simpler meaning representations, such as case-roles (e.g., Brouwer, 2014; Chang et al., 2006; Mayberry et al., 2009; St John and McClelland, 1990). Crucially, Frank et al. (2009)'s model generalizes to both sentences and message representations that it has not seen during training, exhibiting different levels of semantic systematicity.

With regard to whether the model implements or not a symbol system, the situation vectors defined by Frank et al. (2009) were specifically designed to avoid an internal structure composed by a hierarchical combination of discrete symbols; rather, they can be defined as points in a multidimensional continuous space.

Finally, Frank et al. (2009) explain the reason for the development of systematicity to be the inherent structure of the world from which the semantic representations are obtained (similar to Jansen and Watter, 2012). Hence, systematicity does not have to be an inherent property of the cognitive architecture, but rather a property of the representations that are used by the model, where arbitrary symbols with no apparent relation with each other would not suffice. In this way, the model addresses the systematicity issues, and provides an important step in the direction of psychologically plausible models of language comprehension.

This chapter investigates whether the approach developed by Frank et al. (2009) can be successfully applied to the case of language production. Using the message representations described in Chapter 3, the sentence production model described in Chapter 4 is tested to see if it is able to produce sentences related to message representations for which a particular voice was never seen during training (passive vs active), i.e., exhibiting syntactic systematicity; and further, whether the model is able to produce sentences describing situations related to areas in the semantic space that have never been seen during training, i.e., exhibiting semantic systematicity. The results show that the model successfully learns to produce correct sentences in both cases, demonstrating systematicity similar to Frank et al. (2009). Furthermore, the model is not only able to produce a single novel sentence related to a novel message representation, but also it is typically able to produce all possible encodings related to that message, and that are allowed by the grammar that generated the training and test sets, indicating a high level of systematicity.

As first step, the situation vectors defined by Frank et al. (2009) are used, showing that, as they are, the model is able to generate sentences in a systematic way. Afterwards, the model is tested using belief vectors, which, as explained in Chapter 3, are also derived from the Distributed Space matrix, but using a different dimensionality reduction method, and which show a better performance during the simulations. Then, the behavior of the model is analyzed in order to shed some light into the algorithm that the model implements, and how this explains its systematic behavior.

The structure of this chapter is as follows: section 5.1 describes details concerning the testing conditions. Section 5.2 presents the results and analysis of the output for the case where the model is prompted to produce a single sentence. Section 5.2 presents a similar analysis but for the case where the model has to encode multiple sentences. Finally, sections 5.4 and 5.5 present respectively discussion and conclusion of this chapter.

For brevity, in the rest of this chapter, the term “situation” is used instead of “DSS representation”, reflecting the intuition that each DSS representation conveys information about a situation in the microworld.

Some results and parts of this chapter have been presented in Calvillo et al. (2016).

5.1 Testing Conditions

In order to asses the performance of the model in terms of accuracy and generalization, a 10-fold cross-validation schema was employed.

The dataset described in Chapter 3 is a set of pairs $\{(DSS_1, \varphi_1), \ldots, (DSS_n, \varphi_n)\}$, where each DSS_i corresponds to a situation or belief vector, plus a bit indicating if the pair is related to active (1) or passive (0) sentences; and where φ_i is the set of sentences that convey the event related to DSS_i in the expected voice. Then, each testing instance corresponds to giving the model a DSS_i as input and see if the model is able to produce one of the sentences in φ_i.

This dataset contains 782 unique DSS representations or situations in the microworld. For testing, these were divided into two sets: the first one (setAP, $n = 424$) corresponds to those associated to both active and passive sentences, and the second one (setA, $n = 358$) corresponds to situations that are related only to active sentences. The situations in setA correspond to events in which the object of the action is either a person (e.g. “Heidi beats Charlie.”) or undefined (e.g. “Charlie plays.”), and for which the grammar of the microlanguage defines no passive sentences.

The testing conditions are outlined in Table 5.1. The first set (setAP) allowed for three different testing conditions:

- **Condition 1:** Situations for which the model has seen only active sentences, and a passive is queried.
- **Condition 2:** Situations for which the model has seen only passive sentences, and an active is queried.
- **Condition 3:** Completely new situations (not seen during training), passive and active sentences are queried.

	SetAP			SetA	
Condition	**1**	**2**	**3**	**4**	**5**
Known	act	pas	-	act	-
Query	pas	act	act/pas	pas	act/pas

Table 5.1: Testing Conditions

The second set (setA) allowed for two different testing conditions:

- **Condition 4:** Situations for which the model has seen only active sentences, and a passive is queried.
- **Condition 5:** Completely new situations (not seen during training), passive and active sentences are queried.

These conditions represent different levels of generalization or systematicity. In all cases, the queried sentence type has never been seen by the model. For conditions 1, 2 and 4 the model has seen the situations but not in the queried voice. Importantly, for conditions 3 and 5, the model has never seen the situation itself. Finally, for conditions 4 and 5, where passives are queried, not only the model has not seen the passive sentences, but also they are not defined by the grammar of the microlanguage.

We should note that the production model has no access to the grammar that generated the sentences, during training it only has access to semantic representations and their related sentences. Nonetheless, the fact that the grammar does not define passive sentences for the situations in setA means that within the training sentences there are no examples of sentences whose syntactic structure would correspond to the passive sentences that are sought. In other words, for these situations, not only the specific sequences of words related to the sought sentences are new, but also their syntactic structure has never beeen encountered.

According to the 10-fold cross-validation schema, setAP was randomly shuffled and split into 10 folds of 90% training and 10% testing situations, meaning per fold 382 training and 42 testing situations. For each fold, the testing situations were further split into the 3 conditions, rendering 14 different testing situations per condition, per fold.

SetA was also shuffled and split into 10 folds, but in order to preserve uniformity, for each fold and for testing, 14 situations were drawn per condition; meaning that each fold contained 28 testing and 330 training situations.

Finally, for condition 1, the situations were coupled with their corresponding active sentences and incorporated into the training set (while the passive sentences remained in the testing set); vice versa for condition 2. Similarly, for condition 4 the active sentences were incorporated into the training set, while during testing the model will be queried for a passive construction, even though there is none according to the microlanguage.

5.1.1 Sentence Level Evaluation

For a given situation DSS_i, the model is expected to produce a sequence of words $\hat{s_i}$ constituting a sentence describing the state-of-affairs represented in DSS_i. Because a DSS_i can be described by one or more sentences, we assume that the output of the model is perfect if the sentence produced $\hat{s_i}$ is part of the set φ_i of all possible realizations of DSS_i in the queried voice.

However, sometimes the output of the model $\hat{s_i}$ for a DSS_i does not perfectly match any of the sentences in φ_i. As such, we also compute the similarity between the output of the model $\hat{s_i}$, and each sentence in φ_i. This similarity is derived from their Levenshtein distance Levenshtein (1966); which is the number of insertions, deletions or substitutions that are needed in order to transform one string into another. More formally, Levenshtein similarity $sim(s_1, s_2)$ between two sentences s_1 and s_2 is defined as:

$$sim(s_1, s_2) = 1 - \frac{distance(s_1, s_2)}{max(length(s_1), length(s_2))} \tag{5.1}$$

where *distance* is the Levenshtein distance. This similarity measure is 0 when the sentences are completely different and 1 when they are the same. Alternatively, these scores can be regarded as percentages, where 1 corresponds to 100% similarity. Thus, for each item i in the training and test set, we obtain a similarity value:

$$sim(\hat{s_i}) = \max_{s \in \varphi_i} sim(\hat{s_i}, s) \tag{5.2}$$

		Situation Vector (150-dim)		Belief Vector (44-dim)	
Cond.	Query	Perfect Match (%)	Similarity (%)	Perfect Match (%)	Similarity (%)
train	-	87.52	96.89	97.38	**99.22**
1	pas	70.00	92.83	92.86	**98.80**
2	act	69.28	92.14	92.14	**98.36**
3	act	65.71	91.08	90.00	**97.29**
3	pas	70.00	93.67	90.71	**98.24**
5	act	37.14	83.43	87.14	**95.36**
Average Test		62.43	90.63	90.57	**97.61**

Table 5.2: Similarity scores for each test condition.

5.2 Single Sentence Encoding

For each type of semantic representation (150-dimensional situation vectors and 44-dimensional belief vectors), 10 instances of the model were trained and tested, corresponding to each fold as described above. Each instance was initialized with a different set of weight matrices. The scores reported below are averages over these instances.

5.2.1 Quantitative Analysis

We begin by inspecting the performance of the model using the 150-dimensional situation vectors, which can be seen in columns 3 and 4 in Table 5.2. On the training set, the model achieves an average similarity score of 96.89% (with 87.52% perfect matches). This shows that the model is able to learn to transform a situation vector into a sequence of words describing the state-of-affairs that the vector represents.

Regarding the test conditions, on average the model can generate perfect sentences for almost two thirds of the situation vectors. While this performance may seem modest, one should take into account that the 150-dimensional situation vectors went through a dimensionality reduction process, after which some information is lost, in particular, aspects regarding adverbial modification. Because of that, many of the sentences produced contain a wrong modifier such as “well” or “badly”, thus, reducing the number of perfect sentences and similarity scores in general. Nevertheless, on average the model achieves a 90.63% similarity score on the test conditions. Considering that all sentences generated are novel to the model, this demonstrates that the model is able to generalize.

For condition 5 where active sentences are queried, we can notice a drop of performance. This could be explained because setA in general contained fewer sentences per situation, and thus fewer training items. For conditions 4 and 5 where passives are queried but there are no example sentences defined by the grammar, no similarity scores can be computed and in exchange a qualitative analysis will be presented in the next subsection.

The performance of the model using belief vectors can be seen in the last two columns in Table 5.2. In all cases the model performs significantly better with these representations. During training the model achieves a similarity score of 99.22% (with 97.38% perfect matches). Regarding the test conditions, the average similarity score across all of them is 97.61% (with 90.57% of perfect matches). This translates into roughly 1 to 3 errors per condition and per fold. The nature of these errors is addressed in the next section, however we can observe that the performance in terms of similarity is very high and almost perfect in several cases, demonstrating a high level of semantic systematicity.

One should note that considering the training regime, the only hyperparameters that could be optimized are the settings of the learning rate and the number of hidden units. However, modifying these parameters did not yield a better performance: increasing or decreasing the initial learning rate would only change the speed to which the model would converge; and similarly, using more or less hidden units did not represent a significant change, 120 hidden units are used here to mimic the settings of Frank et al. (2009), nonetheless, the results using 80 hidden units are very similar. Consequently the difference of performance between using belief vectors and situation vectors cannot be attributed to the settings of the training regime; in fact, during previous experiments the model was optimized looking for the best hyperparameters using both types of representations independently, obtaining very similar results to the ones reported here.

5.2.2 Qualitative Analysis

Looking at the particular sentences produced by both types of representations, we can notice that in both cases the sentences generated are, with only a few exceptions, syntactically correct and in all cases their semantics are, if not entirely correct, at least closely related to the intended semantics.

Nonetheless, as mentioned before, the dimensionality reduction technique used to generate the 150-dimensional situation vectors introduces some information loss, in particular, information regarding adverbial modification. The sentences containing modifiers do not discriminate properly between “well” and “badly”, and between “with ease” and “with difficulty”. The errors elicited for 3 folds were manually analyzed, finding that during training, out of 445 sentences that contained errors, 209 (46.96%) contained an error regarding modification. Similarly, on the testing conditions, out of 82 items that contained errors, 34 (41.46%) contained an error regarding modification. The information loss affects also other aspects, causing other types of errors, but with fewer attestations.

Almost half of the errors elicited for the 150-dimensional vectors were related to modification, which is an aspect for which no errors are elicited using belief vectors. Thus, we can attribute a big part of the quantitative difference of performance between the two types of representations as arising from the information loss related the 150-dimensional situation vectors. If we do not take into account the errors caused by this, the output obtained using the two kinds of representations is qualitatively similar. Because of that, and since the errors that are elicited using belief vectors are much fewer, we will focus the rest of the analysis on the output obtained using belief vectors. However, we would expect that the performance using the 150-dimensional situation vectors would be similar to the one using belief vectors if the dimensionality reduction did not introduce information loss.[1]

Although the performance obtained using belief vectors is quite high, the model elicits a number of systematic errors that provides us with some insight into the internal mechanism that is implemented by the network. Examples of these are shown in Table 5.3.

The vast majority of the elicited errors occurs when the model produces a sentence that is semantically highly similar to the one expected, reproducing the error patterns of the speech error literature (e.g., Meringer and Mayer, 1895). This pattern can be seen already during training, where the errors arise for situations that are closely related, such that the model is unable to distinguish them, even though it has already seen the situations with their respective sentences (examples 1-3 in Table 5.3).

Table 5.3: Examples of representative output errors.

	Output	**Expected**
1	someone plays with a ball outside.	a girl plays with a ball outside.
2	someone loses in the bedroom.	someone wins in the bedroom.
3	a girl loses to someone in the bedroom.	someone beats a girl at a game in the bedroom.
4	Sophia beats Heidi with ease at hide_and_seek.	Sophia beats Heidi with ease at hide_and_seek in the bedroom.
5	Sophia wins with ease at a game in the street.	Sophia wins with ease at a game outside.
6	a girl plays with a doll inside.	Heidi plays with a doll inside.
7	a game is won with ease by a girl in the bathroom.	a game is won with ease by someone in the bathroom.
8	someone plays.	someone plays with a toy.
9	Charlie plays a game in the street.	Charlie plays in the street.
10	someone wins in the bedroom at hide_and_seek.	someone loses in the bedroom at hide_and_seek.
11	Heidi loses to someone in the bedroom at hide_and_seek.	someone beats Heidi in the bedroom at hide_and_seek.
12	Sophia beats someone at hide_and_seek in the bedroom.	someone loses to Sophia at hide_and_seek in the bedroom.

Sometimes the model produces sentences giving correct information but omitting some details (underspecification). Some other times the model produces sentences that assume information that is not given by the situation, but that it is likely to be the case given the representation (overspecification). One could debate whether these should be considered errors, given that people are not as precise in their productions, sometimes being vague and some other times making assumptions under uncertainty.

In this analysis, we refer as "error" to any difference between the semantic information of the input representation and the semantic information contained in the sentence produced by the model. In some cases, this difference is very subtle and can only be perceived by knowing very specific details of the microworld and the microlanguage (e.g., items 11 and 12 in Table 5.3).

For the conditions shown in Table 5.2, the errors elicited during 5 folds were manually inspected in order to see their regularities. The errors observed (35 in total) can be

[1] See Venhuizen et al. (2018) for an alternative dimensionality reduction technique that may well mitigate this information loss.

classified into 2 main categories: the first one (68.57%) being errors concerning over and underspecification, and the second one (28.6%) corresponding to situations that because of the design of the microworld are remarkably similar, differing only in one aspect.

Concerning the first category, the errors can be further split into **location** under- (14.28%) and over- (5.71%) specification (examples 4-5 in Table 5.3), **subject** under- (22.86%) and over- (11.43%) specification (examples 6-7 in Table 5.3), and **object** under- (8.57%) and over- (2.85%) specification (examples 8-9 in Table 5.3). We can also notice that most of the errors in this category correspond to underspecification (45.71%), corresponding to sentences that express correct information but perhaps not all of the information conveyed in the semantic representation. Again, one could argue whether these can be considered errors or not, taking into account that people often underspecify their utterances.

The errors contained in the second category (examples 10-12 in Table 5.3) correspond to sentences that at first glance seem correct, it is only after taking a deep look into the microworld that one can see the error. First, according to this microworld, whenever there is a winner, there is also a loser, which means that sentences that are apparently contradictory ("someone loses." vs "someone wins.") actually have the same implications within the microworld and therefore are semantically identical. Second, in general whenever there is a winner/loser, the loser is usually situated in the same location as the winner. However, only for the game hide_and_seek and when the participants are inside, the loser can be in the bedroom, while the winner could be in the bathroom, or vice versa. Finally, whenever there is a prepositional phrase ("in the bedroom"), it is attached to the subject of the sentence according to the grammar, which means that in "Heidi beats Sophie in the bedroom", Heidi is in the bedroom while Sophie could be in either the bedroom or the bathroom, while in "Sophie loses to Heidi in the bedroom", it is Sophie who stays in the bedroom while Heidi could also be in the bathroom. Apart from this detail, the situations are almost identical.

More than two thirds of the elicited errors were related to over- and underspecification, where the model produced sentences highly related to the semantics but giving either more or less information than expected. The remaining third was also related to highly similar semantics, were the model was unable to distinguish them. Then, we can conclude that the nature of these errors is related to situations that are highly similar, reproducing relevant findings of the speech error literature.

With regard to the test conditions 4 and 5 where a passive sentence is queried but the grammar does not properly define its characterization, Table 5.4 presents examples of output sentences and the situations that they were supposed to portray. As mentioned before, these situations can be of two types: the first one involving a winning/losing situation where both actors are explicitly mentioned, and the second type being situations where the object of the action is not defined. In order to have a closer view, the output for these conditions and for 5 folds (140 situations) was manually analyzed. The results of such analysis are presented below.

Even though in condition 4 the model has not seen the type of sentences that are queried, and that in condition 5 the model has no experience with the queried situations, the sentences produced by the model are mostly correct and coherent with the semantic information that is given to it. One can see that some information is omitted, but this is expected since the model received no examples of sentences whose syntactic structure would permit the encoding of these situations.

In general, the model learns during training that passive sentences start by mentioning the object of the action. Therefore, for each situation it mentions first this object and then tries to describe the rest of the situation.

Concerning winning/losing situations (examples 1-2 in Table 5.4), which conform 90.7% of the analyzed situations, the object is always a game because in the microworld winning/losing can only happen while playing a game. Thus, the model produces the specific name of the game when it is known (e.g. "soccer is..."), otherwise the sentence starts with "a game is...". Then one of the players is mentioned (one omitted) and the rest of the situation is portrayed.

	Output	**Active Sentence**
1	hide_and_seek is won with ease by Heidi in the playground.	Heidi beats Sophia with ease in the playground at hide_and_seek.
2	a game is won with ease by Sophia.	Sophia beats Charlie with ease.
3	a toy is played with.	someone plays.
4	a toy is played with in the playground by Sophia.	Sophia plays in the playground.
5	a game is lost with difficulty by Charlie.	a girl beats Charlie with difficulty in the street.
6	chess is lost by Heidi in the bedroom.	the boy loses to Heidi at chess in the bedroom.
7	sophia is won with difficulty by charlie.	sophia beats charlie with difficulty.

Table 5.4: Examples of passive output sentences for DSSs with no passive examples.

For the case of situations with an underspecified object (9.3%, examples 3-4 in Table 5.4), it is unknown whether the subject is playing a game or with a toy, so the model is forced to choose one. In most cases (61.5%) the model chooses a toy, which seems reasonable because mostly the situations of the underspecified sentences are more similar to situations where a toy is played with. For example, the situation of "someone plays." is more similar to the one of "someone plays with a toy." (99.36% cosine similarity) than to the representation of "someone plays a game." (97.73% cosine similarity).

Similar to the other conditions, errors regarding over and underspecification occur in conditions 4 and 5, but are rare (7.14%, example 5 in Table 5.4). Two types of error that appear only for these situations corresponds to the inversion of the winning/losing relation in game situations (23.57%, example 5 and 6 in Table 5.4), and the mention of the agent at the beginning of the sentence (4.28%, example 7 in Table 5.4).

In sum, one can see from the output that the model is able to take the linguistic elements learned during training in order to characterize situations for which it has no experience, while being as informative as possible. The only difficulty appears to be the distinction of situations that are highly similar. However, the performance of the model is very high in general and even for the sentences that do present an error, the output is largely correct. In addition, the errors that are elicited serve to further demonstrate systematicity, as they are elicited precisely because of proximity/similarity in a semantic space, where representations are close to each other if they represent similar situations.

5.3 Multiple Sentence Encoding

The previous section showed that the model is able to produce a novel sentence for a given situation. Nonetheless, during training most of the situations that are shown to the model are paired with more than a single encoding. Furthermore, it is clear that humans are able to generate a variety of different encodings for a given semantics. Therefore, it is desirable if not compulsory that a model of human language production should be able to do so as well. In the present section we will test such ability, showing that the model is able not only to produce one novel encoding for a given situation, but in most cases it is able to produce all possible encodings that are supported by the linguistic experience of the model.

5.3.1 Activation Threshold

Previously, at each time step the word produced was defined as the one with highest activation at the output layer. At some time steps, however, two or more words can have similarly high activation, such that all may be allowable continuations at that point of the sentence. In order to give the model the opportunity to explore these derivations, we redefine the policy to produce a word. At a given time step, the word(s) produced is/are not only the one with highest activation, but all the words that reach an activation above a certain threshold τ. By following all possible word derivations that comply with this, the model is able to produce multiple sentences for a given semantics.

We can define τ in several ways, in general it should be low enough such that it allows the model to derive the range of possible sentences, but not too low, so as to avoid overgenerations. A given threshold can be evaluated in terms of precision and recall while trying to obtain all and only the possible sentences given a semantics. In these terms, the following variations were tested:

- **Fixed:**
$$\tau = \rho$$

- **Entropy:**
$$\tau = \rho * \mathbb{E}[logp(w)]$$

- **Ratio to Maximum:**
$$\tau = \rho * \max p(w)$$

where ρ is a parameter for a given run that manipulates how strict the threshold should be for all situations, and $p(w)$ is the probability (activation) of word w at the output layer. The first formulation sets a common threshold for all productions. The second one allows τ to change according to the distribution of activation across all words at a specific derivation point. Finally, the third formulation allows τ to change according to the activation of each word compared to the maximum word activation at each point.

These definitions of τ were used in order to produce multiple sentences for each semantic representation DSS_i in the training set, manipulating the value of ρ. For each DSS_i, precision, recall and f-score of the set of sentences generated by the model were calculated, with respect to the set φ_i that is related to DSS_i; thereafter these values were averaged across the semantic representations in the training set. For these calculations, a sentence produced by the model was taken into consideration only if it was a perfect match with one of the sentences in φ_i, consequently discarding all partial matches.

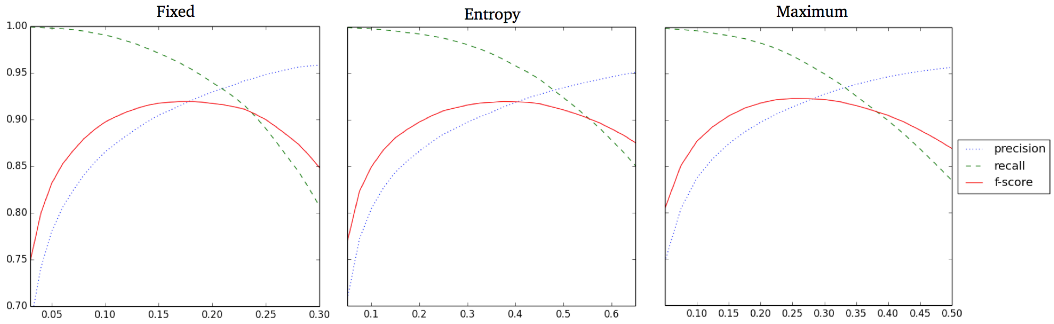

FIGURE 5.1: Precision, recall and f-score for different values of ρ and for different formulations of τ on the training set.

For each of these formulations and for each value of ρ, Figure 5.1 shows average precision, recall and f-score averaging across the 10 folds. We see that the behavior is almost identical between the formulations, although the scales are different. This suggests that the definition of τ is not sensitive to the form of the output distribution, and that a global fixed threshold may be sufficient.

From Figure 5.1, we can also notice that the model achieves a very high recall when setting τ to relatively low values. In this case, precision is rather low, but its value is still high enough to suggest that the model is heavily pruning the derivation forest to only the sentences that are related to the given semantics. Finally, we also see that the maximum f-score value is around 92.5%, meaning that the model is able to a very high degree to reconstruct the whole set of training sentences.

Since the formulation of τ is rather stable, we can set τ as a fixed threshold, and given the shape of the curve, we set it to the value of 0.12, which has a high recall (98%) and a relatively high precision (89%). A high recall value is preferred in order to produce a relatively large number of sentences that could give insight into the production mechanism of the model. This value was used for the rest of the analysis.

5.3.2 Multisentence Evaluation

In this case, the model was evaluated in order to see whether the model was able to produce all and only the sentences that are allowed by the grammar and that are in line with the semantics. The same testing conditions previously defined were used. For each condition and having the activation threshold $\tau = 0.12$, the following was performed: for each DSS_i, precision, recall and f-score values of the set of sentences produced by the model with respect to the set φ_i were calculated; thereafter these values were averaged across the semantic representations of each test condition. For these calculations, only the sentences produced that perfectly matched a sentence in φ_i were considered, discarding all partial matches. Finally, these values were averaged across the previously described 10 folds, giving the results shown in Table 5.5.

Cond.	**Query**	μ**(#sent))**	σ**(#sent)**	**Perfect(%)**	**Precision (%)**	**Recall (%)**	**F-Score (%)**
1	pas	3.34	3.10	59.28	82.16	93.77	84.02
2	act	9.24	11.34	52.14	81.71	96.26	84.92
3	act	7.34	9.64	51.43	78.28	93.40	81.87
3	pas	3.49	2.81	52.86	78.77	91.00	81.36
5	act	8.40	7.25	63.57	83.27	94.13	84.67
Average		**6.91**	**7.13**	**55.86**	**80.84**	**93.71**	**83.37**

Table 5.5: Precision, Recall and F-Score values obtained on the testing conditions.

The first and second columns in Table 5.5 show the test conditions and the type of sentence that the model was supposed to produce. The third and fourth columns show the average size of the set φ_i related to each DSS_i, and the corresponding standard deviation. For example, in condition 2, for each DSS_i the model is expected to produce 9.24 active encodings, with a standard deviation of 11.34. In general, the range is quite wide, where some representations are related to only 1 encoding, while others are related to many more, 130 being the maximum. This variability is presented by the standard deviation values in column 4. On average, the model is expected to produce 6.91 ($\sigma = 7.13$) encodings per semantic representation.

The fifth column in Table 5.5 shows the percentage of situations where the model produced all expected sentences without any over- or undergeneration. In all conditions the model performed perfectly for more than half of the representations (55.86% on average). Considering only these situations, the mean number of sentences per situation was 4.67 ($sd = 5.31$), meaning that the model is able to reproduce a significant number of sentences per semantics without difficulties, in some cases it was able to reproduce up to 40 different encodings without errors. One should also consider that the value of τ was set to have a high recall in order to obtain relatively many encodings, which would give insight into the behavior of the model; but lowering precision, which would ultimately reduce the number of perfect productions.

Columns 6, 7 and 8 in Table 5.5 present precision, recall and f-score accordingly. In all conditions the model produced more than 90% of the expected sentences (93.7% on average), additionally, the produced sentences were mostly correct (80.84% on average). These values correspond to setting $\tau = 0.12$, however, we could vary this parameter in order to obtain higher precision or higher recall, showing a behavior similar to the one obtained with the training set (see Figure 5.1). The value of τ was chosen such that recall would be high, nonetheless, as we will see shortly, the sentences overgenerated, which lower precision, are semantically very similar to the ones expected, which again raises the question of whether they should be considered errors in the first place.

From these results we can conclude, first, that for more than a half of the situations in the testing conditions, the model is able to reproduce without errors all the related sentences, showing that the model can easily generate multiple encodings for a given semantics. Second, from the f-score values we can further say that even for those situations where there are under- or overgenerations, the model is able to reproduce correctly a large proportion of the sentences that were expected for each situation. Finally, we can see in general that the model is able to reproduce almost all sentences that the grammar defines for each given semantic representation, even if the sentences/situations are novel, demonstrating a high degree of systematicity.

5.3.3 Multisentence Error Analysis

Similar to the analysis of single sentence encoding, the errors that the model produces were examined, in order to see if there are specific patterns that would allow us to understand better its internal mechanism.

For a given situation, we can construct a derivation tree of all the sentences that the grammar defines. One can see an example of such a tree in Figure 5.2, where a single semantics is related to 20 different encodings, each one corresponding to a leaf node. In such a representation, S_0 corresponds to the initial state where nothing has been produced yet. Then, each node corresponds to the production of one word. So after S_0, the model has 3 alternatives of word production: “charlie”, “the” (in this microworld there is only one boy, so whenever one says “the boy”, it implies “charlie”) or “heidi”. If the model produces “charlie”, then the alternatives are narrowed down to only “loses”, then to “to” and then to “heidi”. After “heidi”, the model has to choose again between different alternatives, and so on until a period is produced “.”, in which case production halts.

The process of producing all sentences for a given semantics implies the reconstruction of this tree. On the one hand, by adding extra nodes (producing an incorrect word), we obtain overgeneration. On the other hand, by removing some nodes (omitting a correct word), we obtain undergeneration. In both cases, adding/removing one node can potentially result in the addition/removal of several sentences. For this example in particular, the model undergenerated “charlie” as first word, which means that the whole

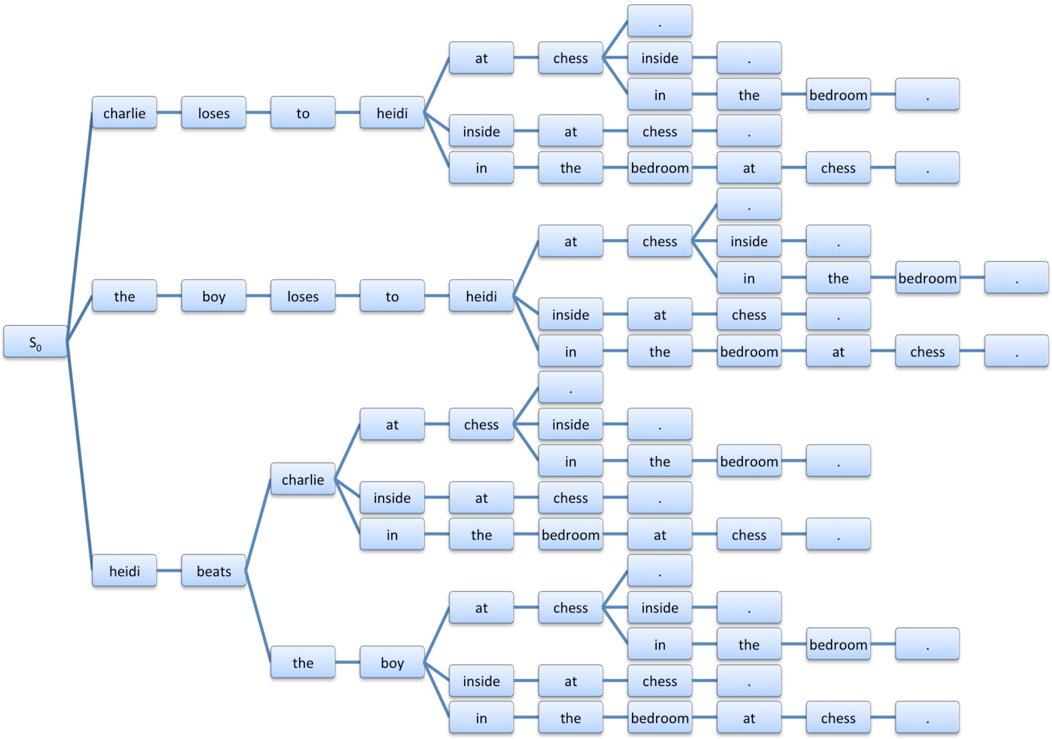

FIGURE 5.2: Example of derivation tree.

first branch was removed, therefore discarding 5 possible encodings. This constitutes a typical undergeneration error.

Figure 5.3 shows an example of overgeneration. In this case, the only sentence that encodes the semantics is “someone plays with a toy.”, however another 3 encodings are produced. First, the model produces “a girl plays”, which allows 2 more different sentences. Then, after “someone plays”, the model produces a period “.”, which truncates the sentence without specifying the object. At the end 4 different encodings are produced, all very semantically related.

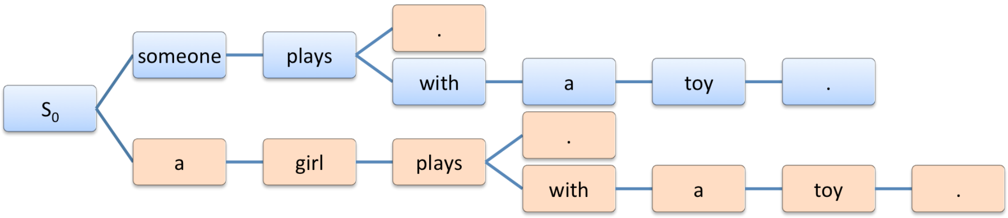

FIGURE 5.3: Example of derivation tree with overgenerations.

5.3.3.1 Manual Analysis of Overgenerations

In order to understand the nature of the errors elicited when the model wrongly produced a word, a manual analysis was made, taking the output of the model for the test conditions and for 3 folds.

In total there were 391 sentences that contained an overgenerated token. These can be split into 4 categories (percentages do not sum up to 100% because some sentences contained more than one error and these pertained to several categories):

	Output	**Expected**
1	a doll is played with in the bedroom.	a doll is played with by charlie in the bedroom.
2	someone wins with ease in the playground.	a girl wins with ease in the playground.
3	charlie plays inside.	charlie plays in the bedroom.
4	a toy is played with by a girl.	a toy is played with by a girl in the bedroom.
5	a toy is played with outside by sophia.	a ball is played with outside by sophia.
6	someone plays.	someone plays with a toy.
7	charlie plays a game in the playground.	charlie plays a game well in the playground.
8	a girl plays a game badly.	someone plays a game badly.
9	a toy is played with inside by sophia.	a toy is played with inside by a girl.
10	charlie plays with a doll in the playground.	charlie plays with a doll.
11	charlie beats heidi with ease at a game in the bedroom.	charlie beats heidi with ease at a game inside.
12	the boy plays a game in the street.	the boy plays in the street.
13	heidi plays with a doll in the playground.	heidi plays with a toy in the playground.
14	heidi loses in the bedroom.	someone beats heidi in the bedroom.
15	heidi loses to someone at hide_and_seek in the bedroom.	someone beats heidi at hide_and_seek in the bedroom.

TABLE 5.6: Representative examples of overgeneration errors.

- **Underspecifying (47.31%):** Sentences that give less specific information than what the semantics contains.

  According to the aspect where the underspecification occurs, these can be split into: subject (33.25%, examples 1-2 in Table 5.6), location (10.99%, examples 3-4 in Table 5.6), object (10.99%, examples 5-6 in Table 5.6) and 2 sentences concerning modality (0.51%, example 7 in Table 5.6).

- **Overspecifying (23.02%):** Sentences that give more information than what the semantics contains.

  Similarly, they can be split according to the aspect where the overspecification occurs into: subject (11.5%, examples 8-9 in Table 5.6), location (5.88%, examples 10-11 in Table 5.6) and object (5.62%, examples 12-13 in Table 5.6).

- **Winner/Loser Location (30.94%):** As described before, these are situations in which hide_and_seek is being played, there is a winner and a loser, and the location of the winner/loser is mistaken due to high semantic similarity (examples 14-15 in Table 5.6).

- **Other (6.64%):** These were 26 errors that did not belong to previous categories (see Table 5.7). Although they are few, they present some interesting phenomena that will be further discussed in the next subsection.

In general some errors are expected, considering that the model explores areas of low probability in order to generate many encodings. Nonetheless, with only a few exceptions, the sentences produced maintain syntactic adequacy and semantic relatedness.

As we can see, the distribution of errors is very similar to the one obtained during single sentence encoding. In this case the errors are also related to semantics that are very close in the semantic space, which perhaps serves to demonstrate the cost of systematicity, as the model processes similar semantics in a similar way.

5.3.3.2 Other Error Types

Amongst the sentences analyzed in the previous subsection, there were 26 sentences that contained errors that did not match the previous categories. Table 5.7 shows examples of such errors. Interestingly, some of these coincide with the ones found in the literature about human speech errors, such as repetitions and substitutions (Clark and Clark, 1980); and only 4 sentences presented a clear syntactic anomaly.

We can classify these errors into 4 categories:

- **Repeated Constituents (13):** Sentences presenting a repetition of one of its segments (examples 1-3 in Table 5.7). With the exception of example 1, these contain 2 prepositional phrases, sometimes complementary (e.g. “outside in the playground”) or repeated with another constituent in between (e.g. “outside with a toy outside”). These show a tendency to produce information as constituents, suggesting some context independence, necessary in order to explain language productivity and systematicity.

- **Game/Toy Substitution (7):** Sentences where “a game” substitutes “a toy” or “a puzzle” (examples 4-5 in Table 5.7). In these cases we can see that the model “knows” that there is similarity between games and toys. In the microworld, games and toys are playable entities, however they are mutually exclusive in terms of the situations in which they appear, with the exception of soccer that is always played with a ball. Additionally, the characters can only win/lose while playing games. Thus, while there is some semantic similarity, we can consider that the model infers their similarity rather based on their linguistic distributional properties.

- **Someone/someone (2):** Sentences of the form “someone loses to someone...” (example 6 in Table 5.7). The grammar of the microlanguage does not define sentences of the form X loses to X, that is, sentences where the loser and winner are the same. However, assuming that “someone” (the winner) is a different person than “someone” (the loser), these sentences do not violate any rule of the microworld, while being intuitively correct.

- **Syntactically Anomalous (4):** Sentences that contain a clear syntactic error (example 7-8 in Table 5.7). These have no apparent reason apart from being derivations of very low probability.

	Expected	**Output**
1	a girl plays hide_and_seek.	a girl plays hide_and_seek hide_and_seek.
2	heidi plays with a toy in the playground.	heidi plays with a toy outside in the playground. heidi plays outside with a toy in the playground.
3	heidi plays with a doll outside.	heidi plays outside with a doll outside.
4	sophia plays with a puzzle in the bedroom.	a girl plays a game in the bedroom. a girl plays a game with a jigsaw.
5	a puzzle is played with by sophia in the bedroom.	a game is played with by a girl inside. a game is played with by a girl in the bedroom.
6	someone loses in the shower at hide_and_seek.	someone loses to someone in the shower at hide_and_seek.
7	a game is played by someone inside.	a game is played by inside.
8	charlie beats sophia at a game with difficulty.	charlie beats sophia at a game with the charlie.

Table 5.7: Examples of other errors.

5.3.3.3 Manual Analysis of Undergenerations

While overgenerations correspond to incorrectly produced sentences, undergenerations correspond to correct sentences that the model chose not to produce. Most undergenerations could be avoided by decreasing the threshold τ, however, these sentences are interesting because they reveal the internal preferences of the model.

Since the omitted sentences are correct, we can only judge these omissions in relation to the sentences that were actually produced. Thus, the sentences that were undergenerated for the same 3 folds as in the previous section were analyzed, comparing them to the ones that were produced. Examples of such sentences can be seen in Table 5.8.

These undergenerations were related to 38 situations, and followed very clear patterns, which can be classified into:

	Undergenerated	**Preferred**
1	the boy plays well in the playground.	the boy plays hide_and_seek well in the playground.
2	a game is won with ease in the bedroom.	a game is won with ease by someone in the bedroom.
3	charlie plays chess well.	charlie plays chess well in the bedroom.
4	sophia plays inside with a puzzle.	a girl plays inside with a puzzle.
5	the boy loses in the street at soccer.	the boy loses at soccer in the street.
6	charlie loses to heidi at chess.	heidi beats charlie at chess.
7	charlie wins at soccer with ease.	charlie wins with ease at soccer.
8	sophia beats heidi in the bathroom.	heidi loses to sophia in the bathroom.

Table 5.8: Representative examples of undergeneration.

- **Overspecification Preferred (28 situations, 43 sentences):** In these cases, while the undergenerated sentences are correct and exact, they leave some parts implicit. For instance, “a game is played in the street.” implies that soccer is the game being played because that is the only game that can be played in the street. However, the model would prefer to say explicitly: “soccer is played in the street.”.

  These behavior can be further split into Object (16 situations, 40 sentences, example 1 in Table 5.8), Subject (9 situations, 26 sentences, example 2 in Table 5.8) and Location (6 situations, 11 sentences, example 3 in Table 5.8).

- **Underspecification Preferred (4 situations, 54 sentences):** In this case, the model prefers to be more ambiguous, leaving details implicit. Nonetheless, this only happens rarely, where the model produces “a girl” instead of “sophia”, “someone” instead of “a girl” and “toy” instead “puzzle/jigsaw” (example 4 in Table 5.8).

- **Order (16 situations, 110 sentences):** The model produced encodings with the same constituents as the undergenerated ones, but in different order. These can be further split into:

  - Location (13 situations, 48 sentences, example 5 in Table 5.8): Location information is preferred to be produced at the end of the sentence.
  - Winner first (3 situations, 62 sentences, example 6 in Table 5.8): The undergenerated sentences were the ones where the loser is mentioned first, showing a preference for mentioning the winner first.
  - "with ease" (1 situation, 6 sentences, example 7 in Table 5.8): There was one case of "with ease", which favored an early attachment.

- **Winner/Loser Location (3 situations, 43 sentences):** Hide_and_seek is played inside, someone wins/loses, and the location of the winner/loser is exchanged. For these situations, the model produced only incorrect sentences, undergenerating the correct ones (example 8 in Table 5.8).

From these results we can distinguish two main tendencies, on the one hand, the model has biases regarding the amount of information that it gives, where overspecification is favored; and on the other hand, the model has specific preferences for constituency order. In both cases, preferences are aligned with the statistical properties of the training sentences.

Regarding overspecification, the related linguistic patterns are more frequent. Concerning games, naming the specific game can be omitted only in situations where soccer is being played in the street or hide_and_seek is being played in the playground, otherwise the game has to be specified in order to avoid ambiguity. Regarding subjects, there are the same amount of sentences where "by someone" is mentioned and where that aspect is omitted. However, taking into account the sentences containing "by charlie", "by sophia" and "by heidi", there are 4 times as many sentences including a phrase containing "by X". Similarly, regarding location, there are twice as many sentences containing a phrase of the form "in the X", rather than "outside"/"inside".

The order of constituents presents a similar behavior. Regarding location, amongst all the sentences that contain "inside"/"outside", for example, only 23.52% mention it in between the sentence, while the rest (76.48%) mention it at the end. Regarding the winner first situations, there are 1572 sentences of the form "X loses to..." and "X loses at...", while there are 4250 of the form "X wins..." or "X beats...". Finally, out of the sentences that contain "with ease", 14.65% mention it at the end of the sentence, while the rest (85.35%) mentions it in between.

In general, one can see that the preferences of the model reflect the statistical properties of the training sentences, as expected and similar to the models presented in Chapter 2 (Chang, 2002; Chang et al., 1997; Dell et al., 1993).

5.3.4 Overgenerations/Undergenerations

The results concerning undergenerations and overgenerations seem to be in contradiction. The undergeneration results suggest that when the model is uncertain, it would prefer to overspecify its sentences. In contrast, the overgeneration results show that there are more errors where an underspecification is elicited than an overspecification. In order to understand this, we consider the structure of the semantic space.

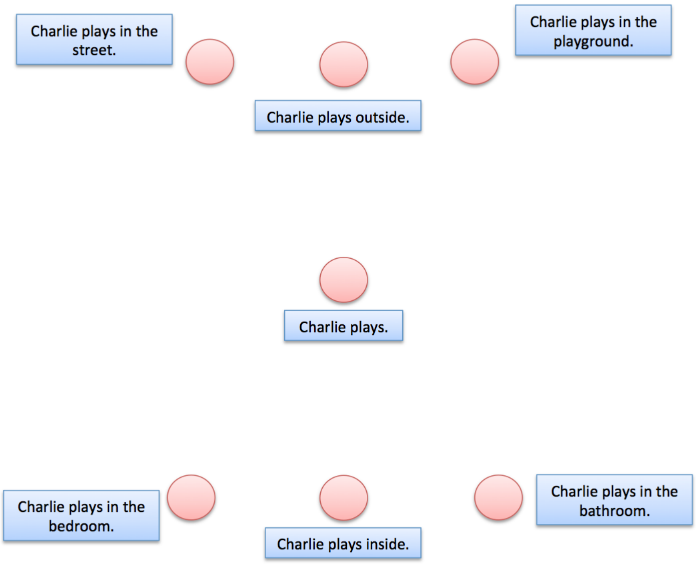
Charlie plays in the street.
Charlie plays outside.
Charlie plays in the playground.
Charlie plays.
Charlie plays in the bedroom.
Charlie plays inside.
Charlie plays in the bathroom.

Figure 5.4: Intuitive example of data points in the semantic space.

The semantic representations that were used are averages over observations where each sentence is true. In Figure 5.4 we can see an intuition of the relative position of different belief vectors in the semantic space. For example, the representation of "charlie plays in the street." is equal to the average of the observations where charlie is playing in the street and likewise for "charlie plays in the playground.". However, the representation for "charlie plays outside." encompasses all the observations in which charlie plays either in the street or in the playground, therefore, the semantic representation of "charlie plays outside." should be in a central position in the semantic space with respect to the other two, as shown in the upper part of Figure 5.4. Similarly, the representation of "charlie plays inside." would be placed in a central position with respect to "charlie plays in the bedroom." and "charlie plays in the bathroom.". Finally, the representation of "charlie plays." is equal to the average among all of the previously mentioned situations, consequently, its location in the semantic space would be at the center of all these points.

Let us consider only the 3 points in the upper part of Figure 5.4. As we saw before, the model has difficulties with situations that are close to each other in the semantic space, so we can predict that the model is likely to have difficulties differentiating “charlie plays in the street.” from “charlie plays outside.”, and similarly “charlie plays outside.“ from “charlie plays in the playground.”. Now, if the model is probed 3 times (once per situation/belief vector), in two out of three times the model could make an underspecification error (producing “charlie plays outside.” when one of the other 2 points were tested); while only in one time an overspecification could be made (when the model is probed with “charlie plays outside.”). Since underspecifying sentences are in general centroids of more specific sentences, there are many more situations where the model is probed to generate sentences that are more specific (similar to “ charlie plays in the street.”). Thus, we should expect fewer overspecifications than underspecifications simply because there are less situations that could generate overspecifications than situations that can generate underspecifications in the dataset.

In conclusion, we can relate overgenerations to states in which the model is confused between different semantic representations, while undergenerations are more related to preferences between sentences that encode the same information. Then, based on the undergenerated patterns, the model prefers to be more explicit while encoding information, following the statistical properties of the training set. However, the model elicits more underspecification errors because there are more situations in the dataset that can elicit underspecification than overspecification.

As we see again, the structure of the input space determines the behavior of the model, including the errors, showing indeed that the model was able to reconstruct this structure and the related linguistic patterns.

5.3.5 Undefined Passive Sentences

In conditions 4 and 5, passive sentences are not defined by the microlanguage for these semantic representations. Therefore, a quantitative analysis is not possible. Rather, a qualitative manual analysis was performed taking the output of the model for the same 3 folds as before.

These representations correspond to areas in semantic space for which the model was never trained to produce passive sentences. As a consequence, the model is more uncertain, such that at each time step more words are activated, producing relatively more sentences. The majority of these sentences is accurate, however, productions with relatively low probability present errors similar to those previously reported.

Generally speaking, the sentences obtained by producing the most activated words are the best, and as one goes further away from the most activated words, errors start to appear, first producing underspecification/overspecification, and then eliciting repetitions or syntactic errors, as word probabilities decrease. Most of these errors could be avoided by setting a higher threshold τ, however these are reported in order to show the internal preferences of the model.

	Active Sentence	**Output**
1	a girl plays in the street.	a game is played with in the street. a toy is played with in the street.
2	someone plays.	a toy is played with by someone. a game is played with by someone.
3	heidi beats charlie with ease outside.	heidi is won with ease by charlie outside. a game is won with ease by charlie outside.
4	charlie loses to sophia in the bedroom.	sophia is lost by charlie in the bedroom. sophia is won by the boy in the bedroom.
5	charlie beats heidi at hide_and_seek inside.	hide_and_seek is lost by heidi inside. hide_and_seek is won by heidi inside.
6	sophia loses to charlie at hide_and_seek in the bedroom.	hide_and_seek is won by sophia in the bedroom.

Table 5.9: Examples of sentences produced for conditions where passive sentences are not defined.

The situations handled in conditions 4 and 5 can be divided into two main categories:

- **Unspecified Object (7 situations):** These situations do not specify whether the object of the action is a game or a toy (e.g., "someone plays", examples 1-2 in Table 5.9). In these cases, the model produces most of the time both sentences where "a game" is mentioned as object of the action and also sentences where "a toy" is mentioned as the object (4 situations). In some situations however, only one option is produced, either "a toy" (2 situations) or "a game" (1 situation). Apart from this, the sentences are in accordance to the semantics.
- **Winning/Losing (77 situations):** These are situations involving two players, one winner and one loser. The microlanguage does not define passive sentences where a person is the object of the action (e.g., "Charlie is beaten."), consequently the model mentions the game that is being played, and then only one of the players, either the winner or the loser (e.g., "a game is lost by Charlie."). Apart from common errors that appear with all situations, for this kind of situation the model elicits errors that can be fit in two categories:
    - Active-like (12 situations, examples 3-4 in Table 5.9): For these situations, amongst the sentences produced there are some with the name of a person at the beginning, mostly replacing the object of the action.
    - Winner/Loser Confusion (39 situations, examples 5-6 in Table 5.9): The model produces sentences where the winner and loser are switched. While

> most of the time the correct role is preferred, for some situations both options are produced.

Concerning the situations where the object is not specified, the model has no way to identify the correct object to produce. Then it is reasonable that both options are produced (“a toy” and “a game”).

About the situations where the active-like sentences are produced, one could explain this as arising also from the uncertainty of the model for this kind of situation. At time t=0, the monitor and context units contain zeros, consequently, the first word production depends only on the semantic representation. Hence, the model activates the words that are semantically related and that could begin a sentence. However, if the model is uncertain, many words would have some, although relatively low, activation. Then, after normalization at the softmax layer, it is possible that semantically related words get an activation beyond τ even if they are not allowed for the current word position.

Concerning the confusion of winners/losers, during training and regarding passive sentences, the model is fed with situations in which there is only one player. This translates into belief vectors in which only one dimension of the type “win(X)” or “lose(X)” is fully activated. Then, it might be that the model learns to detect the kind of situation based on that information, either one in which someone wins or someone loses, and produces the only player in the situation. This strategy would work for situations with only one player, but for situations where both players are included, the belief vectors would contain two dimensions of the type “win(X)” or “lose(X)” that are fully activated, one corresponding to the winner and one to the loser. Then, the model would be uncertain about whether the situation is concerning winning or losing, thus, producing both alternatives, and also producing both participants as possible winner/loser.

In sum, we can see that even though the model exhibits, as expected, some confusion regarding some aspects of the given semantics, it is still able to process most information of each representation. For these test conditions, not only the specific representations are novel to the model, but also the model has never seen this kind of situation coupled with passive sentences. It is only because of the systematic behavior of the model that it is able to produce coherent sentences for these areas of the semantic space. If one takes into consideration the behavior of a classical symbol system under these circumstances, the classical model would be unable to produce any output, as the grammar rules describing passive sentences for these situations are simply non-existent in this microlanguage. In this view, this production model can be regarded as behaving more robustly and perhaps even more systematically.

5.4 Discussion

5.4.1 Semantic Systematicity

Systematicity has been commonly defined as the ability to correctly process novel input instances, generalizing from a set of known ones. This is already widely accepted to be learnable to some degree by neural networks, and is the basis for current computational systems of computer vision and natural language processing, among other applications that are based on deep learning methods (e.g., LeCun et al., 2015). The question we try to address here is that of semantic systematicity, which is the ability to map a correct and possibly unseen meaning representation to an appropriate novel sentence. This behavior is argued to be a sign of human level systematic behavior Hadley (1994a,b). If a connectionist model shows semantic systematicity, it would mean that connectionist architectures are capable of showing a human level systematic behavior, and thus, that connectionist models can be plausible models of human cognition.

In this chapter, this issue is approached from a production perspective: for a given novel meaning representation, produce novel sentences. As we saw in the previous sections, the proposed model indeed learns to generate sentences from the rich situation representations described in Chapter 3, and most importantly it is able to generalize to novel sentences and situations.

First, we can see that the model is able to learn the syntactic patterns of the microworld and does not just memorize sentences, thus showing *syntactic* generalization. This was observed in all test conditions, where the model was able to generate new combinations of words in such a way that the new combinations are in line with the syntactic patterns of the microlanguage, while at the same time being coherent with the semantic structures to which they are related.

Crucially, the model also generalizes *semantically*, as demonstrated in test conditions 3 and 5, where the model was fed with semantic representations that it had never seen, so any correct output can be regarded as arising from the regularities within the microworld from which the DSS representations are derived—cf. the comprehension results by Frank et al. (2009).

Regarding conditions 4 and 5, where a passive sentence is queried but the microlanguage does not define such structures, one can see a behavior that could not be addressed by a classical symbolic model, at least not in an intuitive way. A symbolic model operates over a set of discrete units or symbols, and the operations available are a set of rules that work on these symbols in a predefined and precise way. Then, such a model would find difficulties with items that do not fit into any of the discrete and predefined symbolic units, and furthermore, it would not be able to process combinations of symbols that are not defined by any of the predefined rules. In other words, a symbolic model would have the capability of defining a perfect combinatorial predefined behavior, but would be unable to handle semantic representations for which no specific symbolic rule is defined. The model proposed does not have that issue, it is also able to operate over discrete units, in this case words, but the definition of the semantic space is a continuous one, where unknown areas can still be interpreted. As one can see, the sentences produced by the model for these conditions, are in general semantically and syntactically correct, it is unclear how a symbolic system would behave in this situation.

The fact that the difficulties of the model arise encoding highly similar situations suggests that the model is able to reconstruct the topography of the microworld semantic space, clustering situations that are semantically related. At the same time, the model assigns linguistic structures to each area in this semantic space such that semantically similar situations are assigned linguistically similar realizations. Given that in practice the semantic space is a continuous 44-dimensional space, in theory the model should be able to generate sentences for unseen areas as long as it is given enough information during training in order to reconstruct the semantic space and the mapping between semantics and linguistic realizations, as proposed by Frank et al. (2009).

The results of the test conditions show that this is indeed the case. Conditions 1 and 2 demonstrated that the model is able to generate sentences for semantically known situations but with a different voice (active/passive), showing syntactic systematicity. Conditions 3 and 5 demonstrated that the model is able to generate sentences for unseen areas in the semantic space, thus showing semantic systematicity. Conditions 4 and 5, where passive sentences are queried, demonstrated that the model is able to produce coherent sentences even if the grammar that was used to construct the training/testing sets does not associate passive constructions with these situations. Finally, as we saw during multisentence encoding, the model is not only able to produce a novel sentence for a novel semantic representation, but also it is able to produce most (if not all) of the sentences that are related to the given semantics and that are allowed by the grammar.

5.4.2 Symbol System?

Another argument that has been raised against connectionist models is the following: a connectionist model cannot behave systematically unless it implements a symbol system, if that is the case, then the explanatory value of connectionism is little, and the focus of analysis should be the symbol systems that are implemented on top of it.

One feature of the symbol systems that are mentioned above is that their processing involves the construction of a hierarchical symbolic structure, similar to a syntactic tree using a context free grammar formalism. However, the DSS representations that are used as input for the model presented here do not posses a hierarchical constituent structure, but rather they are better described as points or areas in a continuous space. The fact that the model is able to exhibit systematicity without operating over symbols suggests that a symbol system is not a necessary attribute for systematicity.

Another property of symbol systems is that arbitrary entities can be arbitrarily assigned to symbols, and these symbols are the ones over which the system operates. This process of binding arbitrary entities to symbols is taken for granted in symbol systems, however, the specific details of its algorithmic mechanism or its neural implementation are not clear.

Endowing a connectionist model with this binding mechanism could allow the model to operate with arbitrary inputs. The model of Chang (2002) introduces such a mechanism, where a recurrent neural network essentially operates over semantic role fillers, while an external binding mechanism makes sure that each role is attached to the proper entity. As a result the model is able to produce words in semantic roles that were not seen during training.

While the bindings used by Chang (2002) serve to achieve high degrees of generalization, they are in essence bindings between localist units, and consequently the difficulties assessing similarity between them would also arise. It would be interesting to develop a similar binding mechanism but between distributed representations.

Perhaps such a mechanism is already implicitly in place in some neural network architectures. As we saw before, each layer in a neural network projects the input of the previous layer onto a space that could be regarded as an abstraction of the original. For instance, a layer representing words can be mapped to a layer representing syntactic categories. Then, the only problem is the binding of novel items to the correct areas/symbols, which can be seen as a classification task. Now, it is easy to imagine a mechanism that infers information of a particular novel word from the context in which it appears, in order to assign the word to the correct area in the next level of processing.

This intuition has been shown to be implemented by some neural network architectures. For example, in computer vision using convolutional deep neural networks, it has been shown that each layer contains units that are particularly sensitive to specific features, and that these features become increasingly more abstract with the position of the layer within the architecture (e.g., Yosinski et al., 2015; Zeiler and Fergus, 2014).

Moreover, similar architectures to the one presented here that are used in computational linguistics (e.g., Mikolov et al., 2010; Sutskever et al., 2014) further demonstrate that generalization can be achieved without resorting to the kind of binding proposed by Chang (2002).

In sum, the results of the evaluation demonstrate a systematic behavior of the model, even if the representations used do not possess a hierarchical symbolic structure. Furthermore, the ability to bind symbols to arbitrary entities as in the model of Chang (2002), an attribute that has been proposed to be necessary for systemticity, might already be present to a certain degree in some architectures, as one can see in the generalization results of larger connectionist models that are able to process language in an open domain. These notions suggest that indeed systematicity can be achieved without the need of a full symbol system.

5.4.3 Requirements for Systematicity

An argument that has been posited as fundamental is that while different connectionist models may show systematic behavior, it is necessary to show how systematicity arises a consequence of a neural network architecture and not as a mere coincidence, product of a complicated and tailored training process.

A cognitive system can be defined as a function, where a given input is translated through a set of operations into a specific output. A model trying to learn this function could either memorize each mapping, or could infer the internal computations that the function performs. Since memorization would lead to no generalization, the model would have to infer and approximate the internal process of the function in order to achieve systematicity.

Theoretical analysis have shown that multilayer perceptrons are universal function approximators (Cybenko, 1989). Further, recurrent neural networks have been shown to be at least as powerful as a Turing Machine (Siegelmann, 2012; Siegelmann and Sontag, 1995). Then, the problem of systematicity is not about computational power, but about learning. While a set of connection weights with systematic behavior exists in theory for any given function, the difficulty to learn such weights depends on different factors, such as the complexity of the function to be learned, the representations that are used, or the amount of training data.

The ability to correctly map novel inputs onto their respective outputs implies first, that the outputs depend on the nature of the inputs with some determinism; and second, that the novel inputs share regularities with previously seen items, such that the behavior learned in the past can be applied to the novel inputs. Then, we can recognize factors that could affect systematicity or learnability, some related to the systematicity of the function that the model is supposed to learn, and some other to the systematicity of the input and output spaces and the representations that are used by to the model. A lack of any of these factors would result in lack of learning, regardless of the architecture of the model.

Without discarding the impact that certain architectures have on facilitating learning and/or generalization, the following subsections try to define some conditions about the function to be learned, and the representations that are used, that are necessary and that could index the difficulty for learning a particular behavior. These conditions are defined in the context of connectionist models, but they apply to general learning, assuming that a model trying to learn a given function has no other information than the training set.

5.4.3.1 Structured Input Space

As first condition for generalization is that the input space has to present a structure. If a model with no previous knowledge is trained on a given input set, the knowledge it can acquire is determined only by that set. Then, the information about a given novel input would only be useful if it can be related to those input instances that it has seen before during training. Consequently, the input space has to be to some degree structured, such that regularities can be recognized, learned by the model and applied to new instances.

The degree to which the input space is structured would determine the number of training examples that are needed to reconstruct this space, where more complex spaces would require more training examples. A training set that does not contain enough representative examples of the input space, would not give the model enough information in order to reconstruct the structure of that space. This is similar to the case of inferential statistics, where one tries to generalize the behavior of a complete population given a sample of it. A requirement in such case is that the sample forms part of the population, and that it is representative enough in order to draw conclusions about the complete population.

The events that form the input set of our model and the output set of Frank et al. (2009)'s model are defined in terms of a set of basic events. The space defined by all possible combinations of basic events is structured in the sense that some pairs of basic events always co-occur, some others are not allowed, and other pairs co-occur with some probability distribution. These regularities can be recognized during training, allowing the model to interpolate and infer information about unseen events.

5.4.3.2 Systematic Output

Another requirement for the function to be learned is that it should be systematic. A given function is general or systematic if the same process is applied to all its inputs, whether seen before or not. Then, independently of the number of operations involved, such a function would be at least in theory learnable if its behavior presents regularities with respect to the input space, such that the processing needed for a particular input can be inferred by looking at the processing performed on similar inputs. In other words, similar inputs should be processed similarly.

If that is the case, then for a given novel input, the model would be able to infer the correct processing needed for that particular input, as it should be similar to the process performed on similar inputs in the training set. These regularities would mean also that the output space would present a topology similar or in function of the input space. Then we can speculate that the difficulty of learning a particular function would also depend on the regularity of the output space. For example, we can envision that training a classifier of two linearly separable classes should be far easier than trying to infer the process of a pseudo random number generator.

In our case, the output set of the model is formed by sequences of words that form sentences. These sentences present regularities as they were all constructed using the same grammar, defining a space where some constructions are allowed and some are not. Because of these regularities, the model is able to learn the syntactic patterns of the microlanguage. Furthermore, the mapping between semantic representations and sentences is such that similar semantic representations are processed similarly, as demonstrated by the errors performed by the model. These regularities in the mapping of inputs to outputs permitted the model to process correctly novel inputs.

5.4.3.3 Analogous Informative Representations

We can distinguish between analogous and symbolic representations. Analogous representations depend and show the nature of what is being represented, such that relations between represented items are apparent by looking at the representations (see Frank et al., 2009). In this view, entities that are similar in nature will also have similar representations. Symbolic representations, correspond to arbitrary relations between representations and what is represented, such that the form of the representation is independent of what is being represented.

For example, a photo of a house is an analogous representation of the house that was photographed because it reflects and depends on the nature of that particular house, allowing us to compare that house with other entities by merely looking at their photos; while the phrase “the house” referring to the same house is an arbitrary reference that is only useful if one knows beforehand the mapping between reference and referent.

Assuming that generalization implies the extraction of patterns among known input representations and then using this information in order to process novel inputs, then if there is no link between the nature of what is being represented and the representation, it will also be impossible for any model to draw inferences over the representations of the novel inputs. As a consequence, a necessary condition for the model to learn and generalize is that the representations that are used as inputs contain enough information about the nature of what is being represented, such that this information is enough to discriminate and draw relations with other representations whether known or not.

The situation vectors defined by Frank et al. (2009) and the belief vectors described here are analogous, as their form depends on what is being represented, permitting the comparison of any pair of events just by looking at their corresponding vectors.

Another important and related aspect is informativity. In this case we assume that representations are not perfect reproductions of what they represent, a photo of a house represents the house but it is not the house, consequently not all aspects of the house are available by looking at the photo. Then a representation is informative if it contains the relevant information that is necessary for the task at hand. If for instance, we are interested about the interior of houses, photos from outside would not be very helpful.

The 25,000-dimensional situation vectors defined by Frank et al. (2009) contain very detailed information about the observations in which each event is true. However, after the dimensionality reduction used to create the 150-dimensional situation vectors, some aspects regarding modification are lost, which meant that the model was not able to accurately produce such patterns. In turn, belief vectors do not contain as detailed information as the original 25,000-dimensional situation vectors, only representing averages over observations. In both cases, going from the original 25,000-dimensional situation vectors to belief vectors or to the 150-dimensional situation vectors involved loss of information. Nonetheless in the case of belief vectors, they still contained the required information in order to perform the task at hand, including the aspects of modification.

5.5 Conclusion

In this chapter the language production model proposed in Chapter 4 was tested in order to see first whether it was able to successfully learn to produce sentences utilizing the semantic representations defined in Chapter 3.

The results of the testing conditions showed that indeed that was the case, being able to produce correct sentences for a given semantic representation. An analysis of the elicited errors revealed that the errors were related to highly similar situations, reproducing relevant findings of the speech error literature. Moreover, the errors also reflected the statistical patterns of the training set, showing that the model was able to learn such patterns.

In addition, the model was tested to see whether it was able to exhibit systematicity, being able to generalize and process novel message representations. The results showed that the model able to handle novel message representations. Indeed, the model was able to produce novel sentences and for novel semantic representations. Additionally, the model was able to produce passive sentences for areas in the semantic space for which the microlanguage does not define passive sentences, showing a high degree of systematicity. Furthermore, the model was able to produce not only one sentence but most if not all of the sentences that were related to a particular semantic representation, further demonstrating a systematic behavior.

The results of these tests were partly due to the architecture but more importantly to the semantic representations that were used as input for the model. These are points in a multidimensional continuous space, containing rich information about the situation that a sentence describes.

Finally, a set of conditions were outlined about the nature of the function to be learned and the representations used by the model, in order to learn a particular function showing systematicity. Specifically, the input and output spaces have to be structured, the mapping input-output must be such that similar inputs are processed similarly, and finally, the representations used by a model must reflect the properties of the represented elements. As we saw, these conditions were met by the function and the representations that were used, permitting the model to learn the expected behavior and to exhibit systematicity.

Chapter 6

Sentence Production Dynamics

A Recurrent Neural Network (RNN) is an artificial neural network that contains at least one layer whose activation at a time step t serves as input to itself at a time step $t+1$.

Theoretically, RNNs have been shown to be at least as powerful as a Turing Machine (Siegelmann, 2012; Siegelmann and Sontag, 1995). Empirically they have been shown to be able to learn regular, context-free and context-sensitive languages (Hölldobler et al., 1997; Steijvers and Grünwald, 1996). In computational linguistics they achieve remarkable results in several tasks, most notably in language modeling and machine translation (e.g. Mikolov et al., 2010; Sutskever et al., 2014). In the human language processing literature, they have been used to model language comprehension (e.g. Brouwer, 2014; Brouwer et al., 2017; Frank et al., 2009; Mayberry et al., 2009; Rabovsky et al., 2016) and production (e.g. Chang, 2002; Dell et al., 1993, the results in the previous chapter).

In spite of their success, RNNs are often used as a black box with little understanding of their internal dynamics, and evaluated rather in terms of prediction performance. This is due to the typically high dimensionality of the internal states of the network, coupled with highly complex interactions between layers.

In this chapter, we try to open the black box presenting an analysis of the internal behavior of the language production model presented in Chapter 4. This model can be seen as a semantically conditioned language model that maps a semantic representation onto a sequence of words forming a sentence, by implementing an extension of a Simple Recurrent Network (SRN, Elman, 1990; Jordan, 1986). Because of its simple architecture and its relatively low dimensionality, this model can be analyzed as a whole, showing clear patterns of computation.

The method that was applied is based on Layer-wise Relevance Propagation (Bach et al., 2015). This algorithm is similar to the backpropagation algorithm (Rumelhart et al., 1986). It starts at the output layer and moves in the graph towards the input units, tracking the amount of relevance that each unit in layer l_{i-1} has on the activation of units in layer l_i, back to the input units, which are usually human-interpretable. For a review of this and some other techniques for interpreting neural networks, see Montavon et al. (2017); for related work see Arras et al. (2017); Ding et al. (2017); Kádár et al. (2017); Karpathy et al. (2015); Li et al. (2015).

The analysis, whose details are provided in the rest of this chapter, reveals that the overall behavior of the model is approximately as follows: the input semantic representation activates the hidden units related to all the semantically relevant words, where words that are normally produced early in the sentence receive relatively more activation; after producing a word, the word produced activates syntactic and semantic constraints for the production of the next word, for example, after a determiner, all the nouns are activated, similarly, after a given verb, only semantically fit objects are activated; meanwhile, the recurrent units present a tendency for self-activation, suggesting a mechanism where activation is preserved over time, allowing the model to implement dynamics over multiple time steps.

While some of the results presented here have been suggested previously (e.g., Karpathy et al., 2015), this work represents a holistic integrative view of the internal mechanics of the model, in contrast to previous analyses that focus on specific examples.

The rest of this chapter presents the details of the analysis. The next section gives a reminder of the architecture of the model that was presented in Chapter 4. Afterwards, the model is divided into each of its parts, which are analyzed respectively in each of the following sections. The final two sections correspond to Discussion and Conclusion, which review the main findings.

The results and most content of this chapter were presented in Calvillo and Crocker (2018).

6.1 Sentence Production Architecture

As a reminder, we review the architecture presented in Chapter 4, which is shown in Figure 6.1. The architecture consists of an input layer containing the representation of the message to be conveyed (150-dimensional situation vector or 44-dimensional belief vector), plus one bit indicating if the model should produce an active sentence (1) or a passive one (0); a 120-unit recurrent hidden (sigmoid) layer; and a 43 unit (softmax) output layer where each unit corresponds to a word in the vocabulary.

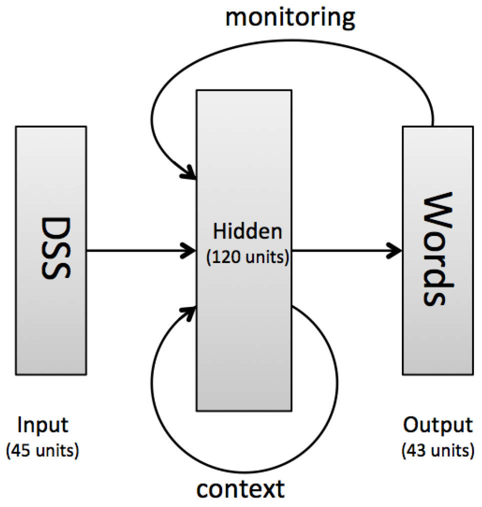

FIGURE 6.1: Model architecture.

Time in the model is discrete. At each time step t, activation of the input layer *dss* is propagated to the hidden recurrent layer. This layer also receives a copy of its own activation h_{t-1} at time-step $t-1$ (zeros at $t=0$) through *context units*. Additionally, the hidden layer receives the identity of the word mon_{t-1} produced at time-step $t-1$ (zeros at $t=0$) through *monitoring units*, where only the unit corresponding to the word produced at time-step $t-1$ is activated (set to 1).

More formally, activation of the hidden layer is given by:

$$h_t = \sigma(W_{ih} \cdot dss + W_{hh} \cdot h_{t-1} + W_{mh} \cdot mon_{t-1} + b_h) \tag{6.1}$$

where W_{ih} is the weight matrix connecting the input layer to the hidden layer, W_{hh} is the weight matrix connecting the hidden layer to itself, W_{mh} is the weight matrix connecting the monitoring units to the hidden layer, and b_h corresponds to the bias unit of the hidden layer.

Then, the activation of the hidden layer h_t is propagated to the output layer, which yields a probability distribution over the vocabulary, and its activation is given by:

$$output_t = softmax(W_{ho} \cdot h_t + b_o) \tag{6.2}$$

where W_{ho} is the weight matrix connecting the hidden layer to the output layer and b_o is the vector corresponding to the output bias unit.

An aspect that facilitates the analysis of this architecture is that the activation of all layers is positive, ranging from 0 to 1. Then, the difference between activation or inhibition of any unit onto another is given by the sign of the connection weight between them. Thus, units inhibiting a particular unit u_i will be those with a negative connection weight to u_i, and activating units will be those with a positive connection weight to u_i.

Having this in mind, the analysis is performed. In this architecture the output layer depends solely on the activation of the recurrent hidden layer. Thus, we will first analyze the influence of the hidden layer onto the output layer, and later we will see how monitoring, input and context units affect production via the hidden layer.

6.2 Word-Producing Hidden Units

As the first step, we would like to know which hidden units are most relevant for the production of each word.

We begin by identifying the hidden layer activation patterns that co-occur with the production of each word. In order to do so, the model was fed with the training set. For each training item, the model was given as input the corresponding message representation, and at each time step the monitoring units were set according to the corresponding sentence of the training item. This process is very similar to one epoch of training, except that no weight updates are made. During this process, for each time a word had an activation greater than 0.2, the activation of the hidden layer was saved. This value was chosen in order to focus the analysis on activation patterns where the target word was clearly activated. At the end, for each word o_k a set of vectors was obtained, each vector corresponding to a pattern of activation of the hidden layer that led to the activation of o_k. Then these vectors were averaged, obtaining a vector that shows which hidden units are generally active/inactive during the production of o_k.

Having these patterns, one can further infer the direction and magnitude of their effect by looking at the connection weights that connect the hidden layer to the output layer.

A hidden unit h_j having a high average activation a_j when producing a word o_k means in general that h_j is relevant for o_k. However, if the weight connecting h_j to o_k is close to 0, then the production of o_k will not be so affected by h_j. In this case, it could be that h_j is only indirectly affecting the production of o_k by activating/inhibiting other words.

Intuitively, hidden units can lead to the production of o_k directly by activating o_k or indirectly by inhibiting other words. Similarly, they can lead to the inhibition of o_k directly by inhibiting o_k, or indirectly by activating other words that compete against o_k. Because of the large number of configurations that can possibly influence production, we will only focus on direct activation/inhibition.

For the case of activation, we obtain a score $A_{h_j o_k}$ conveying the relevance of the hidden unit h_j on the activation of the word o_k, equal to the average activation that o_k receives from h_j when o_k is produced, normalized by the sum of all activation that o_k receives:

$$A_{h_j o_k} = \frac{a_j^k w_{jk}^+}{\sum\limits_{j'} a_{j'}^k w_{j'k}^+} \tag{6.3}$$

where a_j^k is the average activation of unit h_j when the word o_k is produced, and w_{jk}^+ is the positive weight connecting h_j to o_k. This score is only defined for hidden units with a positive connection weight to o_k, which we call activating units.

Inhibiting hidden units are units with negative weights to a word o_k. For inhibition the average activation of an inhibiting hidden unit during the production of o_k is expected to be close to 0. Then, the connection weight is irrelevant, as the product would be close to 0 as well. Thus, for inhibition we do not take into account the average activation, but rather its complement. That is, for each hidden unit h_j, with average activation a_j, we obtain $1 - a_j$ and multiply it by the corresponding connection weight. The result gives us the relevance regarding inhibition of each hidden unit on a particular word:

$$I_{h_j o_k} = -\frac{(1 - a_j^k) w_{jk}^-}{\sum\limits_{j'} (1 - a_{j'}^k) w_{j'k}^-} \tag{6.4}$$

Based on these definitions, for each hidden unit activation/inhibition relevance scores were obtained for each word in the vocabulary. This gives us an idea of the function of each hidden unit. Examples for some hidden units are shown in Figure 6.2, where columns represent hidden units and rows are words in the output layer. The first 5 columns show a sample of the relevance patterns in general, while the rest were chosen because they show some kind of specialization. With the exception of Figure 6.5, the words (rows) in these heatmaps are ordered intuitively according to syntactic and semantic similarity, having in order: determiners, nouns related to persons, nouns related to objects/games, nouns related to locations, verbs, adverbs, prepositions and the period.

One can see that the model takes advantage of redundancy and context sensitivity, where hidden units activate many different words depending on the context. As a result, production of a specific word depends on the combined behavior of all the hidden units, where a word is produced if it receives support from several units.

Nonetheless, some units suggest a specialization (see also Karpathy et al., 2015), activating/inhibiting related words: there are units related to games (e.g., 0, 80), toys (e.g.,

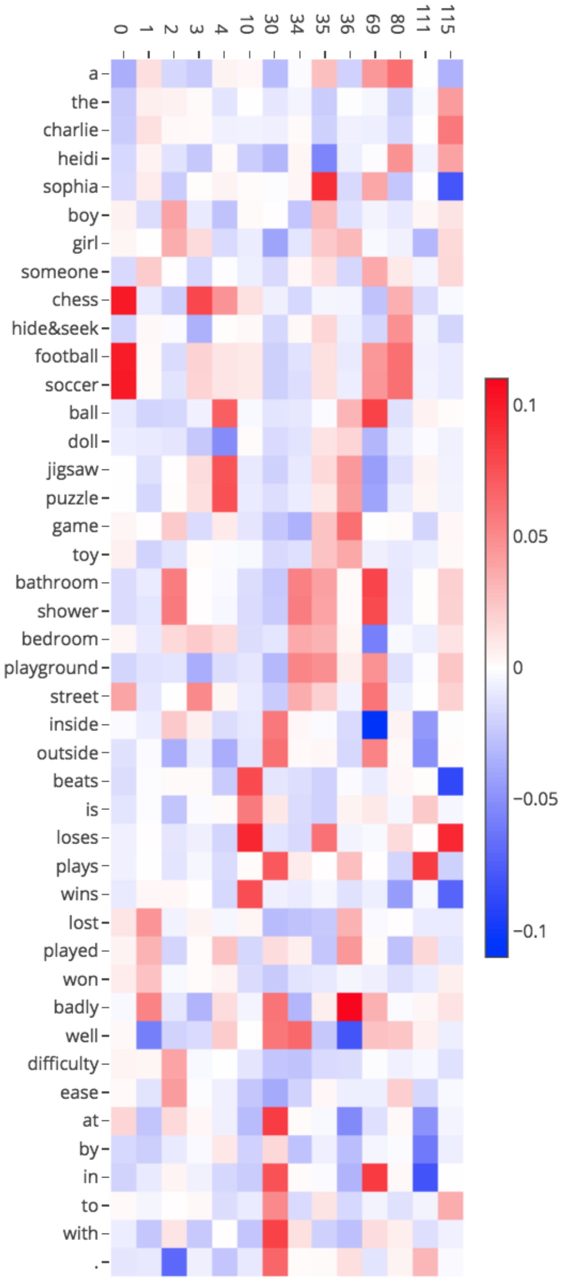

FIGURE 6.2: Relevance scores of some hidden units on output units. Red represents activation, blue inhibition.

4, 36), places (e.g., 30, 34, 35, 69), people (e.g., 35, 115), winning/losing (e.g., 10, 115), prepositions (e.g., 30, 36, 111) and adverbs (e.g., 36).

One can also see that similar words have similar relations with the hidden neurons, suggesting syntactic/semantic categories. A clear example are synonyms, with almost identical relevance patterns, as shown by the rows corresponding to football/soccer, jigsaw/puzzle and bathroom/shower.

While relevance scores of hidden units may be difficult to interpret, they provide a proxy through which one can see the effect of input, monitoring and context units on the output layer, which will be addressed in the next subsections.

6.3 Monitoring Units

Having the relevance values of the hidden layer, one can infer the influence that monitoring units have on the production of each word by looking at their influence on the hidden layer.

The monitoring units feed the hidden layer with the identity of the word produced at the previous time step, where only the unit related to that word is activated (set to 1). Consequently, the effect on the hidden layer depends only on the connection weights of that word. Then, total relevance R_{ik} of the monitoring unit i on the output unit k, is given by:

$$R_{ik} = \sum_j w_{ij} R_{jk} \tag{6.5}$$

where w_{ij} is the weight connecting the monitoring unit i to the hidden unit j, and R_{jk} is the relevance score of the hidden unit j onto the output unit k, which can be activation $(A_{h_j o_k})$ or inhibition $(I_{h_j o_k})$.

Having this, one can further separate and normalize, giving activation A_{ik} and inhibition I_{ik}:

$$A_{ik} = \frac{R^+_{ik}}{\sum\limits_k R^+_{ik}} \tag{6.6}$$

$$I_{ik} = -\frac{R^-_{ik}}{\sum\limits_k R^-_{ik}} \tag{6.7}$$

Figure 6.3 presents these scores. In general, each monitoring unit promotes the activation of words that are allowed after it. Determiners activate the possible nouns that can follow them: “a” activates all toys, “game” and “girl”; and “the” activates “boy” and all locations. Nouns referring to people (e.g., “charlie”, “girl”) activate all present tense verbs and the adverbs “inside” and “outside”. Games and toys activate “is”, in order to form passive constructions. Given that locations appear always at the end of the sentence, they activate the period “.”. Verbs activate words that can serve as their complements, for example “beats” activates all person-related nouns. Similarly, prepositions activate all their possible complements.

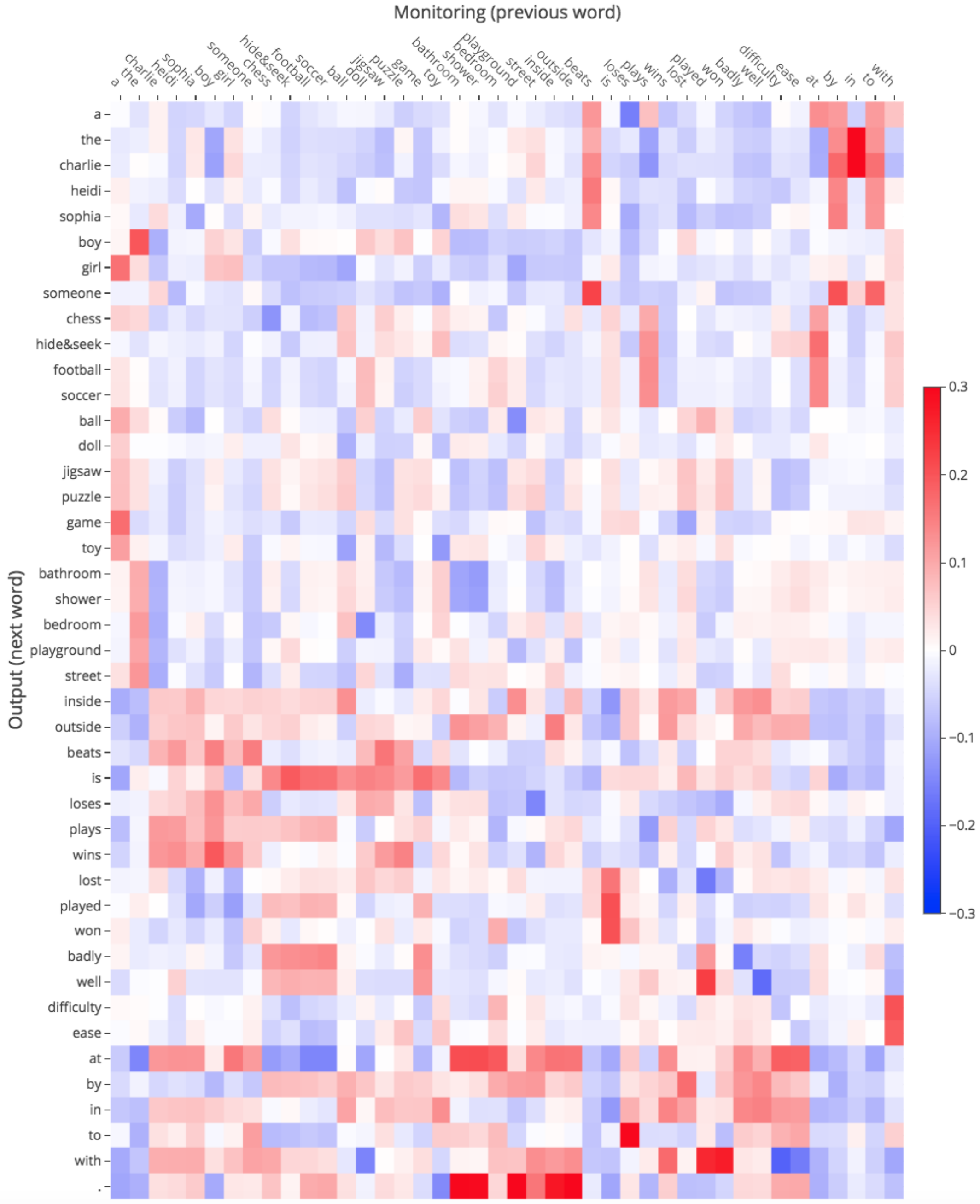

Figure 6.3: Relevance scores of monitoring units on output units. Red represents activation, blue inhibition.

Inhibition works very similarly, where monitoring units inhibit words that should not follow them. For example, determiners inhibit all prepositions, nouns inhibit other nouns as two nouns never occur together, prepositions inhibit also other prepositions, etc. Finally, some words inhibit themselves avoiding repetitions, for example “well” and “badly”.

In general we can see that the monitoring units enforce patterns related at least to bi-grams in the training set, with possibly more long distance dependencies introduced via context units. This can be further verified in Figure 6.4, which shows the activating relevance scores of the monitoring units (above), and the conditional probability of the production of each word given a previous one in the training sentences (bi-gram probabilities, below). As one can see, the main tendencies of the bi-gram probabilities are present in the relevance values, while introducing more information possibly related to dependencies that go beyond bi-grams. We should also note that the relevance scores permit the estimation of graded inhibition relations, something that is not possible using only probabilistic information.

6.4 Input Units

Similar to the analysis performed to the monitoring units, the input units were analyzed in order to see their effect on each output unit.

Using equation 6.5, activation and inhibition scores for the input units were computed, where i would be in this case the index of each input unit. In contrast with monitoring units, many input units can be active simultaneously. Because of that we would like to infer not only the direction of their effect, but also its magnitude in relation to other input units. Hence, the normalization introduced by equations 6.6 and 6.7 was skipped. In this case activation and inhibition correspond respectively to positive and negative values of R_{ik} in equation 6.5. The resulting scores are shown in Figure 6.5.

In general, the input units activate words that are related to their semantics. For example, the input unit $play(sophia, soccer)$ activates words related to sophia, soccer and places where soccer is played (in the street). Similarly, the input unit *manner(win,difficulty)* activates “beats”, “difficulty” and “with”, which are used to convey this aspect. At the same time, each input unit inhibits words that are in conflict with its semantics. For example, the unit $play(charlie, hide\&seek)$ inhibits words concerning other games and the place where that game is not allowed (in the street). This behavior of activation and inhibition can be seen to some degree in all input units.

Of special interest is the last input unit (*actives* in Figure 6.5), which marks whether the model should produce an active or a passive sentence. When this unit is active, words concerning people are activated (e.g., “charlie”, “someone”, “heidi”); at the same time, this unit inhibits words concerning games and passive constructions (e.g., “chess”, “hide&seek”, “soccer”, “is”, “by”). Thus, production of active/passive constructions seems to be determined by giving relatively more activation to words related to people

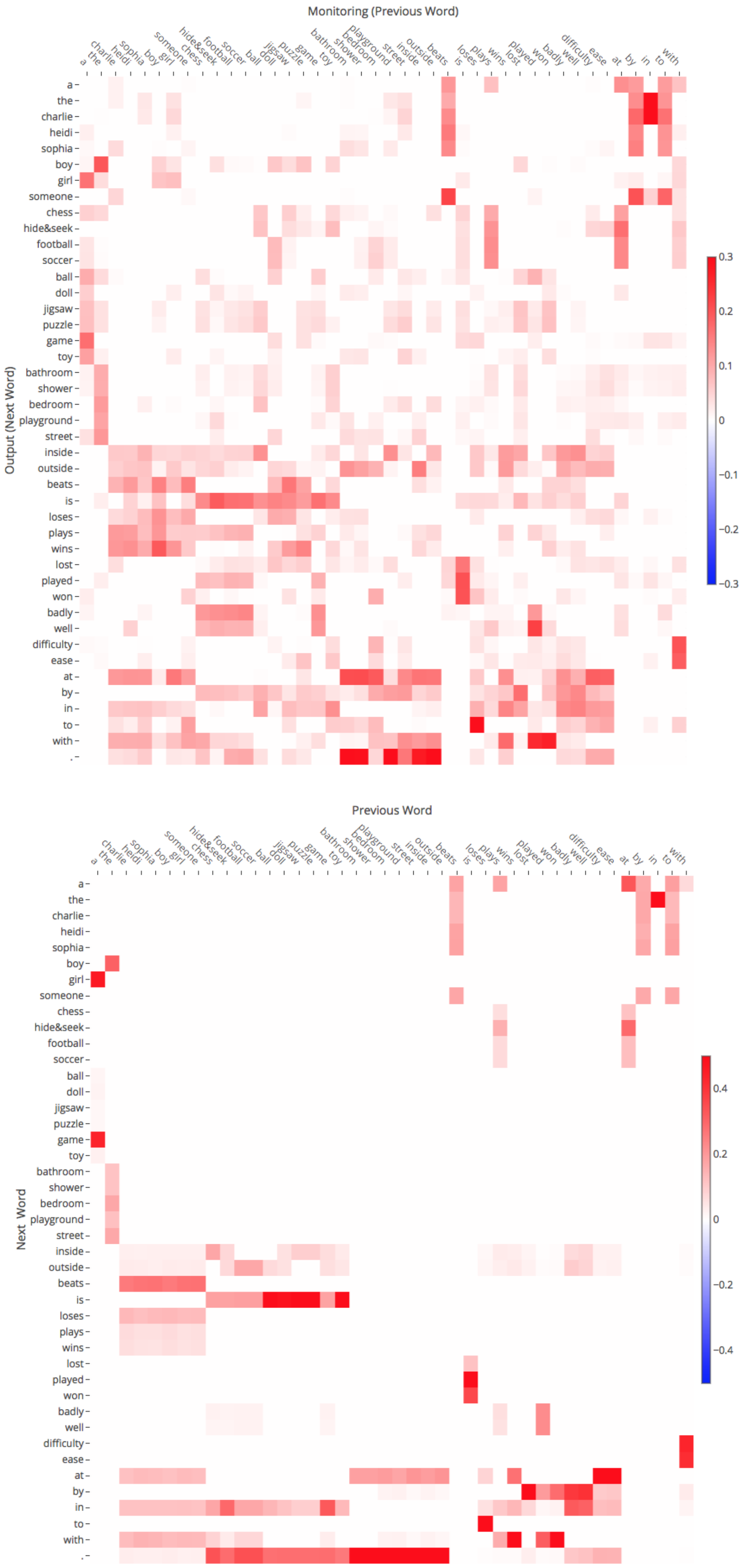

FIGURE 6.4: Above: Activating relevance scores of monitoring units on output units. Below: conditional probability of each word given the previous one in the training sentences.

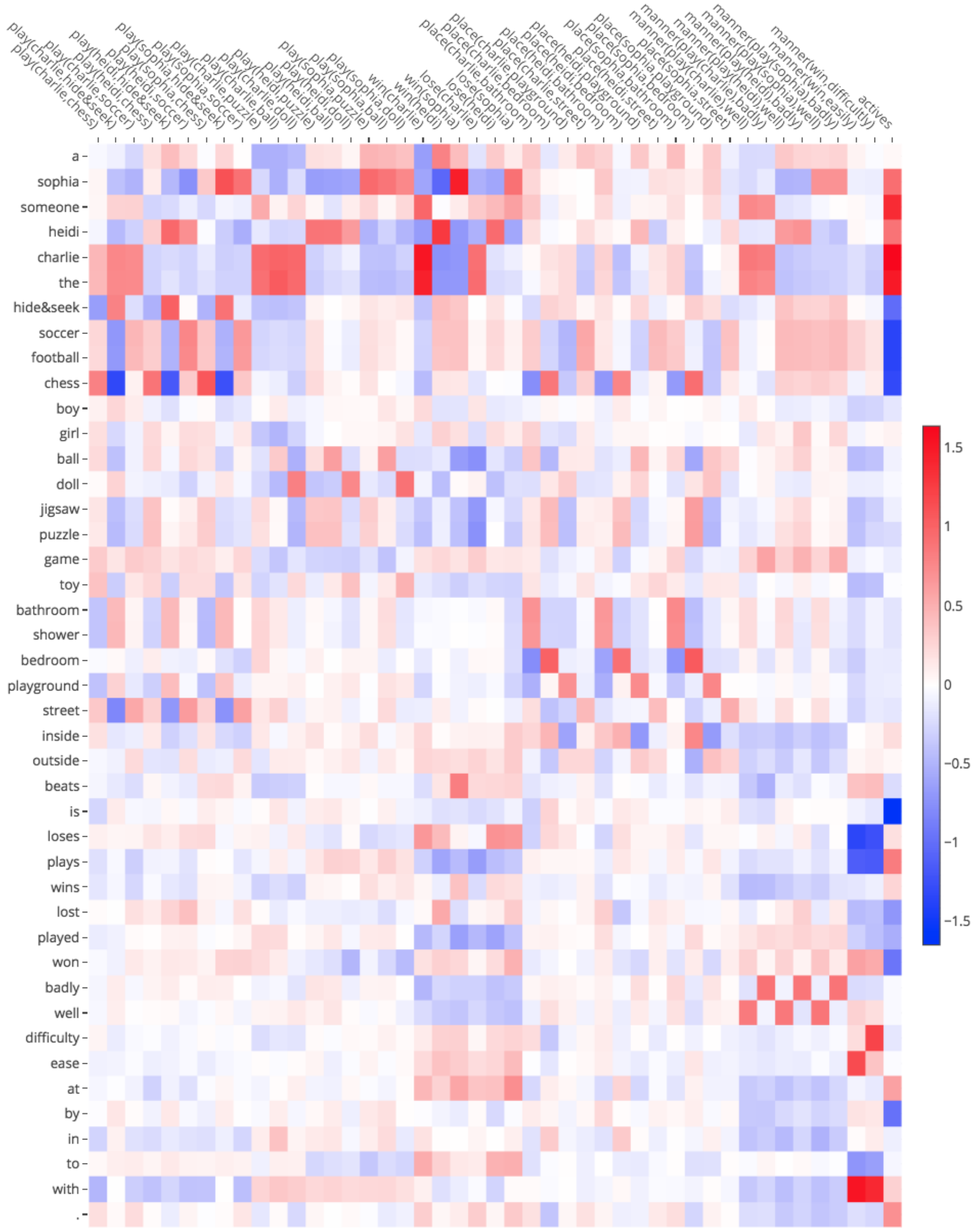

FIGURE 6.5: Relevance scores of input units on output units. Red represents activation, blue inhibition.

in the case of active sentences, or games in the case of passive sentences. This is similar to experimental evidence showing that more conceptually available elements in the message are placed in more prominent grammatical roles (Bock and Warren, 1985; Ferreira, 1994). In the Structural Priming model (Chang et al., 1997) this behavior was implemented by assigning relatively more activation to some parts of the message representation depending on the voice that was to be produced (passive or active). In the model presented here, this behavior was learned by the *actives* unit in the input layer, which as one can see, promotes specific aspects of the message representation depending on the expected voice of the sentence to be produced.

At time step 0, the activation of monitoring and context units is equal to 0. Consequently, the activation of the hidden layer at this point only depends on the input semantic representation. Then, we would expect that the input units should activate more the words that can appear at the beginning of a sentence, relative to other words. This would ensure that the words starting a sentence are correct. Afterwards, monitoring and context units would be able to enforce syntactic and semantic sequential constraints, such that the resulting sentence is coherent.

The words shown in Figure 6.5 are ordered such that the first 10 words are those that can appear at the beginning of a sentence. As one can see, those words seem to receive relatively more activation/inhibition than the rest. Furthermore, the *actives* unit has a very strong relevance, such that when an active sentence is queried, the words that can start an active sentence are more activated.

In sum, the input units influence production by activating hidden units that are related to the semantics that is to be encoded, while additionally giving an idea of the word order that they should follow, specially at time step 0.

6.5 Context Units

At each time step, context units feed the hidden layer with its own activation at the previous time step, providing the model with some kind of memory over possibly unlimited time steps. This is perhaps the most interesting part of the model, however, it is also the most difficult to analyze, given that its content is not directly interpretable, and a relation of causality can involve a variable number of time steps.

We will use the notation h_i to refer to a hidden unit i in the hidden layer, and c_i to refer to the corresponding context unit which contains the activation of h_i at the previous time step.

A way to preserve information over time is by reverberating activation over different time steps. For example, if the hidden unit h_a gets active, then the corresponding context unit c_a will be active at the next time step; if the weight connecting c_a to h_a is such that the activation of c_a causes the activation of h_a, then this would form a cycle in which h_a will be active indefinitely or until other units introduce inhibition, breaking the cycle.

The connection weights between the context and hidden layers were analyzed in order to see if these cycles were present. In such cases, the effect of h_a in the current time step would be similar to the effect of c_a in the next time step. Thus, for each pair (h_i, c_i), if the effect of c_i is similar to the one of h_i, it would mean that c_i is mainly activating h_i or units similar to h_i, forming a cycle. Note that if c_i does not activate h_i directly but other units similar to h_i, it would mean that while the activation of the specific unit might not be preserved, the model would still remain in the same area within the hidden space.

As an example, for the first 15 hidden units Figure 6.6 presents these values. For each pair of columns, the first column represents the direct effect of each hidden unit on the output, identical to the values in Figure 6.2, but normalized for each hidden unit; the second column represents the effect of the corresponding context unit at the next time step, calculated using the equations 6.5–6.7, where in this case i is the index of each context unit.

The column of the right side (DimCorr) presents for all hidden units, the correlations between the relevance values of the hidden units and the relevance values of the context units, related only to each specific word; intuitively showing the degree to which activation related to each word is preserved by all hidden units. The results suggest that the context units tend to preserve activation related to most words, but to different degrees, where activation of words related to toys, locations and adverbs is preserved more than activation of words related to people. Out of the 43 words, 11 presented moderate correlation $(0.4 \leq r < 0.6, n = 120, p < 0.00001)$, and 15 weak correlation $(0.2 \leq r < 0.4, n = 120, p < 0.11)$.

The row at the bottom (UnitCorr) presents correlations between all the relevance values of each hidden unit and the corresponding context unit, that is, between the values of the two columns above. As we can see, some units seem to behave like memory, while others seem to erase their content. For example, units 2, 3, 4, 5, 8, 10 and 14 have a high correlation between the hidden and context relevances, implying a cycle as described above, while units 9 and 11 present an anticorrelation, which means that the context unit is actually inhibiting its corresponding hidden unit. Out of the 120 context units, 14 presented strong correlation $(r \geq 0.6, n = 43, p < 0.00001)$, 26 moderate correlation $(0.4 \leq r < 0.6, n = 43, p < 0.006)$ and 20 weak correlation $(0.2 \leq r < 0.4, n = 43, p < 0.2)$. Regarding anticorrelation, there were 3 units with moderate anticorrelation $(-0.6 \leq r < -0.4, n = 43, p < 0.0036)$ and 6 with weak anticorrelation $(-0.4 \leq r < -0.2, n = 43, p < 0.2)$.

As we can see, about half of the context units have a tendency to preserve their activation, which varies according to each unit, and to the kind of related information. This suggests a tangible mechanism that preserves information over time, which in the case of language is necessary in order to enforce long distance dependencies.

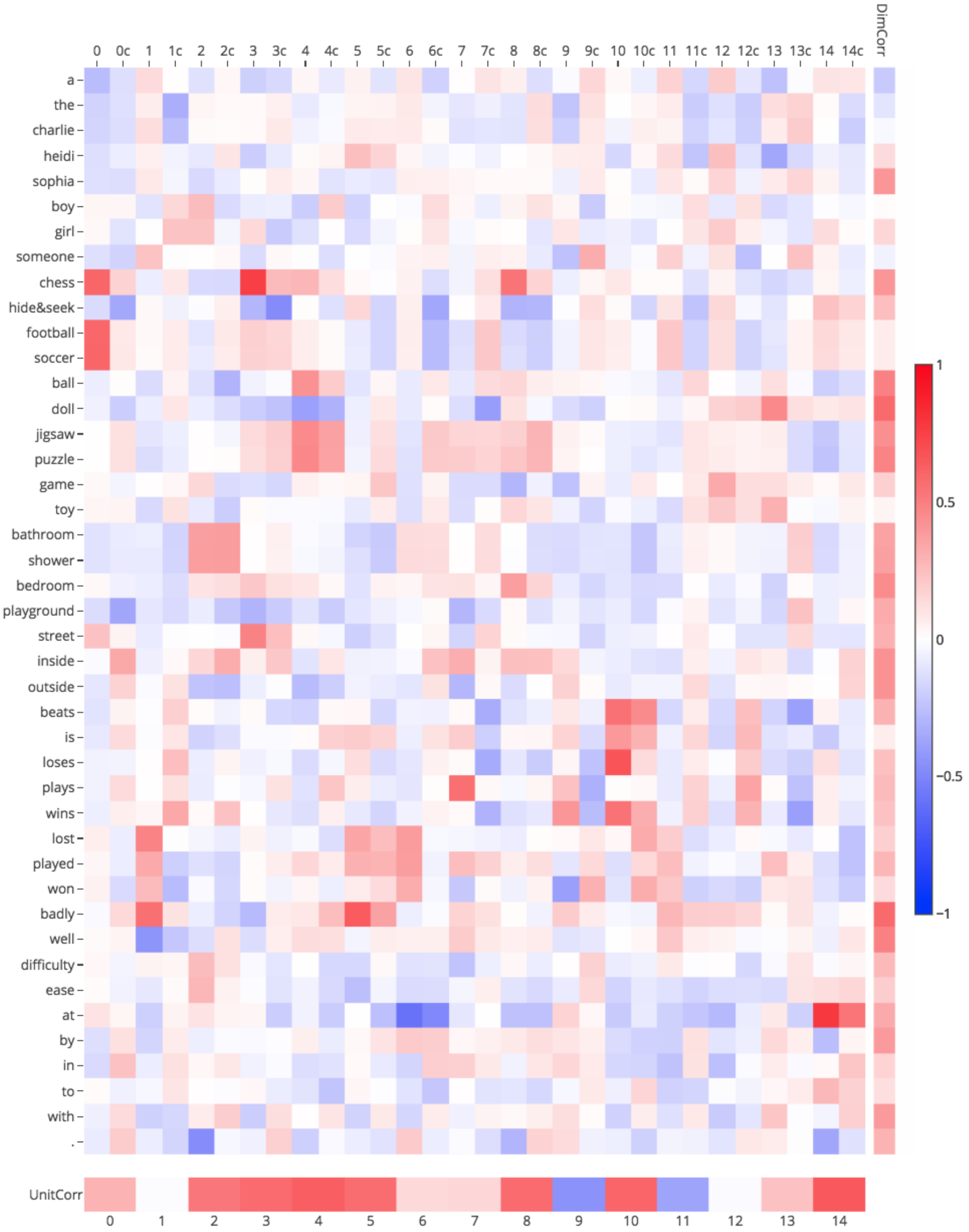

FIGURE 6.6: Relevance scores of hidden and context units. Right: correlations for each word between the relevance values of the hidden units and the context units. Bottom: correlations between all relevance values of each hidden unit and the corresponding context unit.

6.6 Discussion

In the above sections, the language production model was separated into its different modules in order to determine their function. The results show that each layer in the model serves a different purpose in the architecture. Trying to integrate these parts into a global explanation of the internal mechanics of the model, we arrive to the following:

1. Production starts when the model is fed a semantic representation at time step 0. At this point, the semantic representation is the only source of information, as the context and monitoring units are set to zeros.

2. Based on the activation of the input layer, the model must produce a word that is in accordance to the semantics and that is syntactically plausible for the beginning of the sentence. As we saw, the input units seem to select the words necessary for production, and depending on the voice expected (active or passive) more activation is given to the words that can fulfill the first position.

3. After the initial word has been produced, monitoring and context units gain influence. Monitoring units promote the production of words that can follow the previous word, and inhibit words that should not follow. At the same time, context units keep information regarding previous and current activation, suggesting a sort of memory, where information remains latent until the right time to be produced. This happens until a period is produced, in which case production halts.

These patterns are in line with the view of the lexicon as a productive system, and not just a collection of lexical items upon which operations are made (Elman, 2014). Viewed from this perspective, the network is a dynamical system where the model moves within a continuous multidimensional space represented by the hidden layer states. These movements are determined by the semantic representation that serves as input to the model and the lexical transitions that unfold over time, such that each lexical item that is activated modifies the state of the dynamical system moving it in certain directions within the hidden space (see Elman, 1995). As we saw, the model analyzed here exhibits a similar behavior: the semantic representation modifies the state of the hidden layer promoting the aspects that are relevant for the given semantics; meanwhile, at each time step a lexical item is activated, which in turn modifies the hidden state through the monitoring units, introducing syntactic and semantic constraints. In this case, each lexical item in the monitoring layer operates over the hidden states and not the other way around, as described by Elman (2014).

Because of the high architectural similarity, one could expect that the production process presented here might generalize to some of the models described in Chapter 2 that also use a recurrent architecture; particularly, the Phonological Error model (Dell et al., 1993), the Structural Priming model (Chang et al., 1997) and the Prod-SRN model Chang (2002), whose architectures are almost identical to the one presented here. While the Dual-Path model also implements a recurrence, its binding-by-weight mechanism makes more difficult to draw commonalities between that architecture and the one analyzed here.

Some larger models used in computational linguistics possess also a relatively similar architecture (e.g., Mikolov et al., 2010; Sutskever et al., 2014). These models typically employ a higher dimensionality and more complex hidden units such as Long-Short Term Memory (LSTM, Hochreiter and Schmidhuber, 1997) or Gated Recurrent Units (GRU, Cho et al., 2014). Nonetheless, the main paths of computation are largely similar to the ones described here: at each time step the word previously produced is fed to a recurrence that in turn feeds another layer yielding a probability distribution over the vocabulary (e.g., Mikolov et al., 2010); additionally, a semantics is fed into the recurrence in semantically conditioned models, such as some used in machine translation (e.g., Sutskever et al., 2014) or image caption generation (e.g., Chen and Lawrence Zitnick, 2015). Furthermore, the individual results presented here are coherent with previous findings on larger architectures (for example, similar words are known to have similar word embeddings), suggesting that these results can be generalized to such models.

While some adaptation might be needed for specific cases, the algorithm described above might serve as intuition of how those models work, and the methodology outlined here could serve to test such a hypothesis in future work.

6.7 Conclusion

This chapter presented an analysis of the internal mechanism of the language production model that was described in Chapter 4. The results show clear patterns of computation that permit us to infer its internal mechanism.

As we saw, each hidden unit is related to a variable degree to each word in the output layer, such that some hidden units promote the activation of some words while inhibiting some others. In turn, the input semantic representation constantly promotes the activation of areas in the hidden space that are related to words that are relevant for the expression of the given semantics. Regarding the monitoring units, they enforce syntactic and semantic constraints of words sequences. Finally, the context units preserve information over different time steps, and thereby are able to enforce long distance dependencies.

Because of architectural similarities with other language models, we expect that this mechanism reflects the behavior of similar models models of language production (e.g., Chang, 2002; Chang et al., 1997; Dell et al., 1993), as well as larger models used in computational linguistics, such as those used in language modeling (e.g., Mikolov et al., 2010) or machine translation (e.g., Sutskever et al., 2014), among others. The methodology here outlined could also serve to test such a hypothesis in future work.

Chapter 7

Approximating UID

For a given semantics, humans are able to produce a large number of utterances that express its meaning. However, some constructions are preferred over others, some sentences are easier to understand, while some others are more difficult, so people tend to avoid them.

As described in Chapter 2, the Uniform Information Density Hypothesis (UID, Jaeger, 2010; Levy and Jaeger, 2007) presents one way to rank sentences according to how uniform their surprisal profiles are, where a sentence is preferred if the surprisal of each of its words is more uniform than for alternative encodings. This is proposed as a rational strategy of language production at the computational level of analysis, since such a strategy maximizes the probability of successful communication in a bandwidth-limited noisy channel while maximizing information transmission. Alternatively, and without the assumption of a noisy channel, comprehension effort is also minimized utilizing a UID strategy (Levy and Jaeger, 2007), provided that the effect of surprisal on comprehension effort is superlinear (Hale, 2001a; Levy, 2008).

Empirical evidence supports this hypothesis (e.g., Aylett and Turk, 2004; Bell et al., 2003), however, we are aware of no modeling attempts that explore this at the algorithmic or implementational levels. In this chapter, a mechanistic account of sentence production is presented, which balances on the one hand the rate of information transmission and on the other hand comprehension and production effort. The sentences produced by this strategy present more uniform surprisal profiles, compared to other strategies, and thus, offer a first algorithmic approximation to UID.

In particular, the model assumes that, while achieving the goal of producing sentences that convey the intended message, speakers act under three different pressures: a first one, pushing speakers to be fast under time restrictions; a second one, related to production effort, pushing speakers to produce available content first (see Ferreira and Dell, 2000); and a third one, related to comprehension effort, pushing speakers to avoid high information density structures. This chapter describes a way to balance these pressures in order to obtain sentences with more uniform surprisal profiles, which could be later linked to a bandwidth-limited communication channel.

The results and most content of this chapter have been published in Calvillo (2017).

7.1 UID Model

The language production model proposed here extends the one described in Chapter 4. Its architecture, shown in Figure 7.1, consists of two paths of processing: the first one (above, inside the dotted rectangle), computes word probabilities given the context, and is identical to the model described in Chapter 4; and the second one (below), receives the output of the former and computes derivation length estimations, i.e., how long a sentence can be if a particular word is produced. We call *probabilities* the layer containing the output of the first path, and *der_lengths* the layer containing the output of the second path.

The output of these two paths is then combined in a final layer (*words*) that receives unmodified copies of the activation of *probabilities* and *der_lengths*, and whose activation is a combination of these two types of information. At this point the model produces the word with the highest activation in *words*, whose identity is then passed to the first hidden recurrent layer through monitoring units in order to process the next word production. Finally, production stops when an end-of-sentence marker is produced.

The rest of this section presents in more detail each of these parts, along with their justification.

7.1.1 Semantic and Linguistic Information

The information content or *surprisal* of a sentence s is defined as its negative log probability $-logP(s)$. Moreover, sentences express events or situations in the world, such that a sentence can be paired with one or more events, and vice versa. Therefore, we can decompose the probability of a sentence s into:

$$P(s) = \sum_i P(s, e_i) \tag{7.1}$$

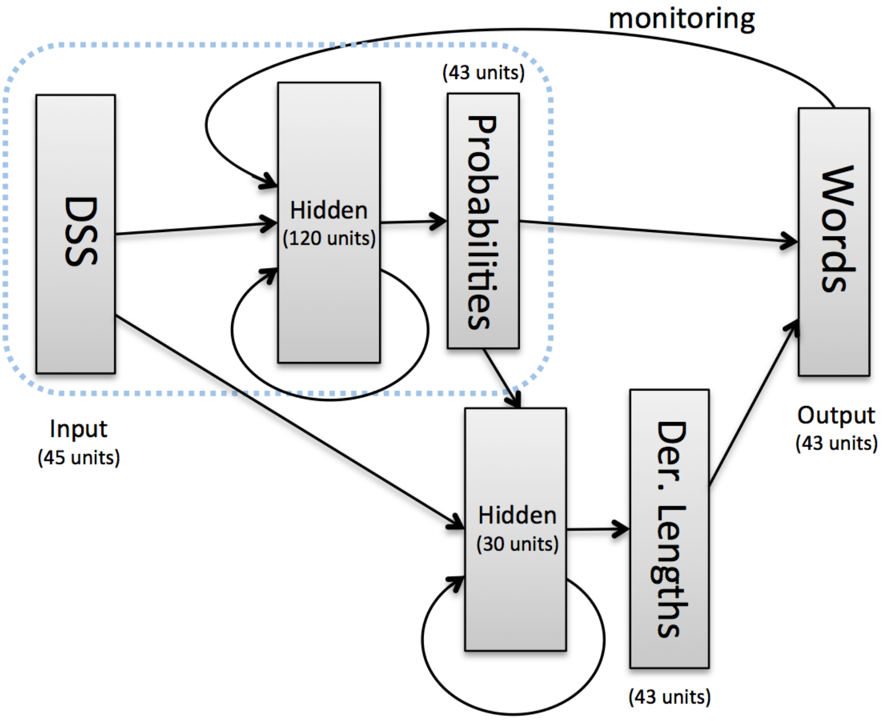

FIGURE 7.1: UID Production Model.

$$= \sum_i P(s|e_i)P(e_i) \tag{7.2}$$

where e_i is each event in the world that is paired with s (e.g., if 5 events are paired to a particular sentence, i would range from 1 to 5).

From this, we can distinguish two kinds of information: $P(e_i)$, related to each event that can be paired with the sentence; and $P(s|e_i)$, related to the linguistic elements used in this particular sentence to express e_i.

We call the first one *semantic* surprisal, and the second one *linguistic* surprisal. Semantic surprisal represents how unexpected the events conveyed by the sentence are. Linguistic surprisal can be seen as the information that the sentence conveys, when the semantics is *already known*; thus, it is not information about the world, but about the sentence itself.

Frank and Vigliocco (2011) use a similar terminology for similar concepts. Like them, we use the term *semantic surprisal* to refer to $-logP(e)$, which is the information related to an event in the world. Nonetheless, they use the term *syntactic surprisal* referring to $-logP(s)$, while we use the term *linguistic surprisal* referring to $-logP(s|e)$.

These two types of information, semantic and linguistic surprisal, cannot be easily disentangled because they are embedded in each sentence/event. Knowing the identity of an event gives information about the possible related sentences, and vice versa. Nonetheless, based on our definition, we can express total semantic surprisal of a sentence s as:

$$SemSurp(s) = -log\sum_i P(e_i) \tag{7.3}$$

where e_i is each event that can be expressed by s.

While one sentence can be paired with several events, normally when a speaker produces a sentence, he/she has one specific event in mind e_α. For example, for the sentence "the girl sees the man with the telescope.", we can distinguish two different interpretations: one where the girl is using a telescope to see a man, and another one where a girl is seeing a man who carries a telescope. While the sentence is ambiguous, the speaker would already have a concrete situation model in mind while producing the sentence, which can be either of the interpretations, but not both. Similarly, while a word might have multiple meanings, when a speaker produces a word, he/she usually refers to one single meaning.

To differentiate the meaning to which the speaker refers from other possible interpretations, we use e_α. Then, while total semantic surprisal is as described above, the semantic information/surprisal that the speaker is trying to communicate is only:

$$-logP(e_\alpha)$$

As a result, the relevant information associated with a specific sentence s assuming that the speaker is trying to communicate the event e_α is given by:

$$Surp_{e_\alpha}(s) = -logP(s|e_\alpha)P(e_\alpha) \tag{7.4}$$

$$= -logP(s|e_\alpha) - logP(e_\alpha) \tag{7.5}$$

where the semantic information $-logP(e_\alpha)$ remains constant across all different surface realizations that could convey it; in contrast to the linguistic information $-logP(s|e_\alpha)$, which can vary widely depending on the specific syntactic structures or words that the speaker chooses.

7.1.2 Being Easy to Produce

Surprisal Theory (Hale, 2001a; Levy, 2008) states that the cognitive effort associated to the processing of a word is proportional to its surprisal. Evidence supporting this has been shown for comprehension (e.g., Hale, 2001a; Levy, 2008), and production (e.g., Jescheniak and Levelt, 1994). Therefore, one can assume that a rational model of production would try to minimize effort for both interlocutors.

While comprehension effort is minimized when speakers follow a UID strategy, production effort can be minimized by following an Availability Based Production strategy (ABP, Ferreira and Dell, 2000), where items are produced as they are available. Assuming that more probable words are also more available, then producing the most probable word at each time step minimizes (to some extent) production effort by locally minimizing linguistic surprisal:

$$w_{t+1} = \arg\min_{w} -logP(w|DSS, w_0, ..., w_t) \tag{7.6}$$

where w is a word in the vocabulary and DSS is the semantic representation related to e_α. This is already implemented by the model described in Chapter 4, where the word produced at each time step is the one with highest conditional probability given the semantics and the previously produced words. In the model presented in this chapter, these probabilities are obtained at the *Probabilities* layer in Figure 7.1.

Additionally, we can note that such a model produces sentences with low information content by minimizing their linguistic surprisal, and therefore minimizes to some degree comprehension effort.

7.1.3 Being Fast

The information contained by a sentence results from the sum of the information contained by each of its words. Thus, knowing that the semantic surprisal related to e_α should sum up to $-logP(e_\alpha)$, and that this information is distributed among the words in the sentence, we can calculate average word semantic information/surprisal with respect to e_α:

$$E[WordSemSurp_{e_\alpha}] = \frac{-logP(e_\alpha)}{n} \tag{7.7}$$

where n is the number of words in the sentence. Hence, if one wants to maximize average semantic information transmission of the desired event e_α, it suffices to minimize n. We hypothesize that in general speakers tend to maximize information transmission of the desired semantics e_α by minimizing n, and therefore by favoring shorter sentences.

The model presented minimizes sentence lengths by estimating at each time step a score that reflects the expected derivation length that would follow the production of a certain word. This is done by the second path shown in Figure 7.1, below.

This path is constituted by a hidden recurrent layer followed by a softmax layer. The recurrent layer contains 30 sigmoid units and receives as input the DSS semantic representation, the output of *probabilities*, and its own activation at time step $t-1$ (zeros at $t=0$). Activation of this layer is then propagated to a softmax layer (*der_lengths*) with dimensionality equal to the size of the vocabulary(43), and that calculates for each word a score DL that resembles a probability distribution, where values closer to 0 represent longer derivations and values closer to 1 represent shorter derivations, and where probability mass is distributed among all words that can be produced at the given time step. Finally, these layers receive also input from a bias unit with a constant activation of 1.

A model that produces at each time step the word that maximizes this score would prefer words leading to shorter derivations, regardless of their information content:

$$w_{t+1} = \arg\max_{w} DL(w|DSS, probabilities_{t+1}) \tag{7.8}$$

Nonetheless, if a model were to follow solely this tendency, the resulting sentences would tend to have high information density because it would choose highly informative words in order to minimize derivation lengths.

7.1.4 Being Easy to Comprehend

Intuitively, a model combining the previous two strategies would produce sentences with more uniform surprisal profiles, compared to a model that only applies one of them. However, these strategies do not take into account that world events with high surprisal represent a higher comprehension effort.

Speakers know beforehand how unexpected the event they are trying to communicate is. Therefore, one can propose that they balance these two strategies according to this information. That is, when a speaker is trying to communicate an event e_α with low surprisal, the speaker would prefer to be faster, minimizing derivation lengths; but, when the event represents high surprisal, the speaker would prefer sentences with lower linguistic surprisal and possibly longer. Thus, at each time step, the model would produce the word that maximizes the score:

$$w_{t+1} = \arg\max_{w}\{(1-P(e_\alpha))P(w|...) + P(e_\alpha)DL(w|...)\} \tag{7.9}$$

This final model is expected to produce sentences with more uniform surprisal profiles, compared to strategies that only maximize one of these measures, or that do not take into account semantic surprisal.

In the model this is computed at the *words* layer (see Figure 7.1), which receives $P(w|...)$ values from the *probabilities* layer and $DL(w|...)$ scores from the *der_lenghts* layer.

The use of semantic surprisal to balance brevity and minimization of linguistic surprisal could be partially explained in terms of production effort. Since the semantics to be communicated is known by the speaker, no semantic processing is needed from the speaker side, and therefore we could assume that production effort is driven mainly by linguistic information. However, speakers still need to access the lexical information of the words that are needed. For example, we can imagine an event e_β with high surprisal, and that is so specific that only one sentence can encode it. Then, the sentence encoding it s_β would have a conditional probability of 1 given the semantics, and thus, 0 linguistic surprisal. Nevertheless, if the words needed to construct s_β are very infrequent, we can still predict a higher effort in order to retrieve their lexical information.

Assuming that infrequent words tend to have higher semantic surprisal, then we can use semantic surprisal as a surrogate to measure the effort of lexical retrieval during production. Then, under high retrieval effort contexts, minimization of linguistic surprisal is preferred as a way to overall minimize production effort. And likewise, under low retrieval effort contexts, brevity is preferred because the production system still has resources to minimize derivation lengths.

However, this would not be the case for function words such as “that”, which have high frequency and therefore would not represent high retrieval effort. Then, omission or addition of “that” where both options are available, would not be driven by lexical retrieval, but rather by information density in order to reduce comprehension effort, as the UID Hypothesis proposes (Jaeger, 2010).

7.2 Training and Evaluation

7.2.1 Examples Set

The model was trained and evaluated using the dataset described in Chapter 3, which consists of a set of pairs $\{(DSS_1, \varphi_1), \ldots, (DSS_n, \varphi_n)\}$, where each DSS_i is a message representation plus a bit that indicates if the pair is related to active (1) or passive (0) sentences, and φ_i is the set of the sentences that describe the situation that is related to DSS_i and in the expected voice. In this case, the model uses belief vectors, as they showed a high performance during the sentence production task in Chapter 5, eliciting fewer mistakes than when using situation vectors.

This set relates each message representation DSS_i to several sentences, permitting the definition and evaluation of different ranking functions. Nonetheless, the distribution of sentences was not especially designed for the experiments performed, in future work a new set could be defined in order to obtain a more clear setting.

In addition, the model uses two more types of scores that are related to each pair (DSS_i, φ_i), namely, scores related to the length of a sentence after the production of a given word, and scores related to the prior probability of a particular semantic representation. These are explained below.

7.2.1.1 Derivation Length Scores

For each DSS representation, it is known beforehand the sentences that can encode it according to the microlanguage. Furthermore, it is known at each derivation point what words can be produced and how long the sentences would be if a particular word is produced. Using this information, we compute a score that resembles a probability distribution over the vocabulary, but that reflects the length of the sentences that one can expect after producing a particular word, such that values close to 1.0 denote relatively short sentences, while values close to 0 denote relatively long sentences.

More specifically, given a DSS representation and a derivation point, for each possible word production w_i, we obtain its minimum derivation length $min\_dl(w_i)$, which is the length of the shortest sentence that can be produced if w_i is produced. Afterwards we calculate a score $dl(w_i)$:

$$dl(w_i) = \max_{w}\{min\_dl(w)\} - min\_dl(w_i) + 1 \tag{7.10}$$

which is equal to the difference between the greatest min_dl value among all the words that can be currently produced and the min_dl associated to each specific word w_i, plus 1. Finally, we normalize by dividing by the sum over all the possible word continuations.

$$dl(w_i) = \frac{dl(w_i)}{\sum\limits_{j} dl(w_j)} \tag{7.11}$$

According to these scores, all possible word productions at a specific derivation point have a positive value that is inversely proportional to the length of the shortest sentence that can be obtained by following that production. In the UID model, these are the values expected as activation of the layer *der_lengths*.

7.2.1.2 Semantic Probability

For each DSS representation related to an event e_i in the examples set, a semantic probability value $P(e_i)$ was also computed. Using the original situation vectors, the prior probability of an event can be calculated by averaging the values of each dimension of the corresponding situation vector, as explained in Chapter 3. However, one should consider that the proportions of sentences and events in the dataset should be in concordance with the event probabilities.

The probability of an event e_i can also be calculated by:

$$P(e_i) = \sum_j P(e_i, s_j) \tag{7.12}$$

where s_j is each of the sentences that describe the event e_i. This formulation suggests that whenever e_i occurs, a corresponding sentence describing e_i is also produced. In a natural setting this is not the case, in fact a large proportion of possible events are never uttered. This necessarily means that in the calculation above, one must include the case where an event occurs without a sentence being uttered, in order to obtain the total probability of that event. Furthermore, given a particular event to be conveyed e_i, some of the sentences that can express the event might be more frequent than others, that is, for each sentence s_j related to e_i, the value of $P(e_i, s_j)$ might be different.

Consequently, in order to obtain a dataset with these characteristics one should on the one hand pair events to empty utterances for the cases where an event is not described linguistically, and on the other hand, the distribution of sentences given an event should also reflect the intuition that some encodings are more common than others, even if they refer to the same semantics.

Unfortunately, because of time restrictions when the experiments were conducted, such a dataset was not built. Rather, the original dataset was used, and therefore, the semantic probability values were computed accordingly. Considering that the model is trained only on the pairs given in the examples set and that all sentences are presented an equal number of times during training, then the probability of a DSS representation related to an event e_i is given by the number of sentences related to that representation divided by the total number of sentences in the examples set:

$$P(e_i) = \frac{n_i}{N} \tag{7.13}$$

where n_i is the number of sentences that describe the event e_i, and N is the total number of sentences in the dataset. Thus, semantic probability is proportional to the number of sentences related to a given semantics. While these values might seem in conflict with the event prior probabilities that can be obtained through the situation vectors, the belief vectors only contain conditional probabilities, and therefore the notion of prior probability of an event would rather be inferred from the dataset.

Additionally, since $P(e_\alpha)$ is used to balance word probabilities and derivation lengths in the model, less biased values are needed because as it is, $P(e_\alpha)$ is in general very low, while $1 - P(e_\alpha)$ is very high. Therefore instead of normalizing by the total number of sentences N, normalization is done with respect to the highest number of sentences that can be related to a DSS representation, which is 130. Hence, for each DSS, its probability $P(e_i)$, or henceforth $P(DSS)$, is given by the number of sentences paired with the representation, divided by 130.

7.2.2 Training Procedure

Since the output layer receives unmodified copies from *probabilities* and *der_lengths*, the connections from the latter to the former are fixed one-to-one and do not need training. In other words, the i^{th} unit of *probabilities* is only connected to the i^{th} unit of *words* with a connection weight fixed to 1, and likewise for the connections between *der_lenghts* and *words*.

Prior to training, all weights on the projections between layers (with the exception of those mentioned in the last paragraph) were initialized with random values drawn from a normal distribution $\mathcal{N}(0, 0.1)$. Weights on the bias projections were initially set to zero.

Training consists of setting the connection weights leading to the computation on the one hand of *probabilities* and on the other hand of *der_lengths*, corresponding to the two paths of processing. Accordingly, training is performed in two phases, in both cases using cross-entropy backpropagation (Rumelhart et al., 1986) with weight updates after each word in the sentence of each training item. These two phases are explained below.

- **probabilities:** The first phase corresponds to the training of the path leading to *probabilities*, which is performed as described in Chapter 4, where the model is trained to predict the next word given the semantic representation and the previously produced words.

  During this phase, the monitoring units were set at time t to what the model was supposed to produce at time $t - 1$ (zeros for $t = 0$). This reflects the notion that during training the word contained in the training sentence at time-step $t-1$ should be the one informing the next time step, regardless of the previously produced (and possibly different) word. During production, the monitoring units are set to 1.0 for the word that was actually produced and 0.0 everywhere else.

This path was trained for a maximum of 200 epochs, each one consisting of a full presentation of the training set, which was randomized before each epoch. Note that each item of this set consisted of a DSS_i paired with one of the possible sentence realizations describing the state of affairs represented in DSS_i. Hence, during each epoch, the model saw all the possible realizations of DSS_i. An initial learning rate of 0.124 was used, which was halved each time there was no improvement of performance during 15 epochs. No momentum was used. Training halted if the maximum number of epochs was reached or if there was no performance improvement over a 40-epoch interval.

- **der_lengths:** The second path is trained after the training of the first one is completed. During this phase, the connection weights calculated during the first phase are fixed, so that only the second path weights are modified.

  At each time step, the DSS is fed into the first path, which outputs a probability distribution over the vocabulary. This is fed into the second recurrence, as well as the DSS representation. Monitoring units are handled exactly as in the first training phase. The activation of the second recurrence is then propagated to *der_lengths*. Its output is compared to the derivation length values, as defined in the previous section, and finally the connection weights are updated.

  Training of this path was performed for a maximum of 80 epochs, with the training items arranged in the same way as in the previous phase. An initial learning rate of 0.24 was used, which was halved each time there was no improvement of performance during 10 epochs. No momentum was used. Training halted if the maximum number of epochs was reached or if there was no performance improvement over a 20-epoch interval.

7.2.3 Evaluation

The UID model defines a production strategy as an interaction between production goals. In order to assess the behavior of the model, its productions were compared to those obtained by using alternative strategies, where at each time step the model produces the word according to:

1. Minimum Linguistic Surprisal
2. Minimum Derivation Length

3. Maximum (Word Probability +/* Derivation Length Score)

4. Complete UID Model

The third strategy refers to a combination of minimizing linguistic surprisal and minimizind derivation lengths through a sum or a product of the related scores. Complete UID Model refers to the model where linguistic surprisal and derivation lengths are balanced using the semantic surprisal.

For each DSS representation in the examples set that was related to more than one sentence (968), the model generated a sentence according to each production strategy.

In order to measure surprisal, a language model was trained implementing a Simple Recurrent Network (Elman, 1990). This model is constituted by a recurrent hidden layer that receives at each time step the identity of the previous word, as well as its own activation at the previous time step. The activation of this layer is then forwarded to a softmax layer that computes a probability distribution over the vocabulary, given the previous words. This architecture is very similar to the production model presented here in Chapter 4, except that it contains no input semantic representation.

This model was trained on the whole set of sentences for 200 epochs with a learning rate of 0.24 which was halved each time there was no improvement in performance.

Using this language model, surprisal values were calculated for each word w_i of the produced sentences as folows:

$$surp(w_i) = -logP(w_i) \tag{7.14}$$

where $P(w_i)$ is the probability that the model assigned to that word given the context (its activation).

Uniformity of information density was measured in terms of standard deviation of word surprisal, assuming that complete uniformity would produce a standard deviation of 0 across all the sentences that were produced:

$$\sigma_{surp_w} = \sqrt{\frac{1}{N}\sum_{i}^{N}(surp(w_i) - \mu_{surp_w})^2} \tag{7.15}$$

where σ_{surp_w} is the standard deviation of word surprisal, N is the total number of words produced, $surp(w_i)$ is the word surprisal of each indivual word w_i that was produced, and μ_{surp_w} is the mean word surprisal of all the words that were produced.

7.3 Results and Discussion

The results can be seen in Table 2. The first column denotes the production strategy that was used by the model in each case. The second column presents the production performance, similar to the evaluation in Chapter 5, where each value equals the average similarity between the sentences produced by the model and the sentences that describe the corresponding DSS representations in the dataset. The third column shows the average sentence or derivation length (μ_{DerLen}) of the sentences produced by the model. Similarly, the fourth column shows the average word surprisal (μ_{surp_w}). And finally, the last column shows the standard deviation of word surprisal (σ_{surp_w}).

	Accuracy	μ_{DerLen}	μ_{surp_w}	σ_{surp_w}
Minimum Linguistic Surprisal	99.67	9.01	1.0	0.89
Minimum Derivation Length	99.86	**7.55**	1.20	0.97
Max P(+/*)DL	99.82	7.77	1.16	0.95
Max 3P-2DL	98.23	10.15	**0.89**	**0.84**
Complete UID Model	97.67	10.17	**0.89**	**0.83**

TABLE 7.1: Results of each production strategy.

As expected, minimizing linguistic surprisal led to lower word surprisal values and longer sentences, compared to minimizing derivation lengths. Combining these two strategies by a sum or product led to results almost identical to each other, and very close to minimizing derivation lengths, suggesting that derivation length scores were mostly dominating production. Nonetheless, the sentences produced by this naive combination already present slightly higher uniformity and lower average word surprisal, compared to the sentences produced minimizing only derivation lengths.

Given that linguistic surprisal and derivation lengths are different in nature, one can expect a more complex relation between them, other than a sum or product, in order for the resulting score to be helpful. In order to explore slightly more complex relations, a grid search was performed in order to find linear factors that would minimize the standard deviation of word surprisal. The resulting model corresponds to the fourth row in Table 7.1, where the model produces at each time step the word that maximizes:

$$3P(w|DSS, w0, .., w_n) - 2DL(w|DSS, probabilities) \tag{7.16}$$

where one can see that minimizing linguistic surprisal is favored, while minimizing derivation lengths is penalized. As a result the sentences produced are longer than only minimizing linguistic surprisal. However, uniformity of information density is higher than with the previous models and additionally average surprisal is lowest.

The final row in Table 2 presents the results of the model that incorporates semantic probabilities. For this case grid search was also used, which led to a model that at each time step produces the word that maximizes:

$$(3.5 - P(DSS))P(w|...) + (P(DSS) - 2.5)DL(w|...) \tag{7.17}$$

which is very similar to the previous model, but with some influence from semantic probabilities. While the performance of this model is very similar to the previous one, its sentences present slightly higher uniformity of information density; and the influence of semantic surprisal is in the expected direction, where semantics with high surprisal produce longer sentences and vice versa.

Interestingly, while uniformity of information density increases for the last two models, production accuracy decreases, suggesting something similar to Degen and Jaeger (2011), where speakers sacrifice precision in order to accommodate robust communication.

As one can see, maximizing uniformity of information density led to models that produce longer sentences, reducing the efficiency of communication. It is possible that the production strategy previously outlined is the one that maximizes information density given the possible encodings in the language, however, at the cost of production efficiency. Evidently, it is not only necessary to obtain sentences with uniform surprisal profiles, but also, one must specify the channel capacity to which the surprisal profiles should adapt. A proper definition and operationalization of the capacity of the communication channel remains for possible future work.

The small difference between the last two strategies highlighted some issues of the models presented that could cause this behavior, and that will be addressed also in future work:

- First, the nature of the language model used to test the model, which receives no semantic information during training, which means that rather than being a joint model of semantics and sentences, it only considers word sequences.
- Then, the production model proposed here uses semantic surprisal at a sentence level, while speakers can be sensitive to this information incrementally at a word level.
- Finally, the dataset used to train the model did not contain specific contrasts between event and sentence probabilities that could permit the use of semantic surprisal as a strong factor to modify the behavior of the model.

In general the model outlined here shows: first, that as expected, shorter sentences are more dense in terms of information content. Second, that longer sentences can encode information in a more uniform way. Third, that sentences with more uniform information densities present in average lower word surprisal, therefore minimizing comprehension effort. And finally and most importantly, that sentences with higher uniformity of information density can be produced by balancing sentence lengths and word probabilities. In future work, this can help to address uniformity for a given channel capacity.

7.4 Conclusion

This chapter presented a model of language production that takes into account word probabilities and sentence lengths in order to produce sentences with uniform surprisal profiles, and in order to model the Uniform Information Density Hypothesis.

The sentences produced by this model were compared to those produced using other strategies, showing that the proposed model produces sentences with more uniform surprisal profiles and lower average word surprisal, by balancing sentence lengths and linguistic surprisal.

This model represents a first attempt to model the Uniform Information Density Hypothesis at the algorithmic level, where uniformity arises by balancing word probabilities and sentence lengths in a mechanistic way. As first attempt, it requires further work, however, it serves to highlight some issues that need to be addressed regarding UID, namely:

- The difference between comprehender and producer in terms of information processing. During communication, the producer already knows the message to be conveyed, therefore most of his/her effort would be focus on the retrieval of lexical items and the construction of syntactic structures. In contrast, the comprehender has to infer the message, disambiguating incrementally according to the information given by the sentence. In such case, the effort would be related to the recognition of the linguistic input, as well as the construction of a proper message representation. Consequently, one can speculate that speakers would rather focus on the processing of linguistic surprisal, building linguistic structures; while comprehenders would emphasize the processing of semantic surprisal, building a message representation.
- In order to properly train a UID model as the one proposed, one needs a proper dataset that reflects in a naturalistic fashion the relations that sentences and events

have, such that the probability of producing a sentence can be related to the probability of the corresponding event. This is important because this information is learned implicitly by the model, so any related and expected behavior needs to be present in the dataset during training.

- In order to test the model measuring surprisal from the point of view of a comprehender, it is necessary not only to measure the probability of the specific sequence of words, but also, one should also consider the probability of the related events. For example, a very common syntactic structure utilizing very common words, would represent low linguistic surprisal; however, it might convey an event that hardly ever occurs, meaning a high semantic surprisal. In such case, a language model that has no notion of world events, would not consider the effort that a comprehender would need in order to process such an utterance.
- Finally, the definition and quantification of the capacity of the communication channel is necessary in order to properly define efficient communication. As seen in the results, one can obtain very uniform surprisal profiles by selecting words whose probability is close to 1, carrying very little information and resulting in very long sentences. Perhaps those long sentences can be considered efficient in environments where the capacity of the communication channel is very low, while in some other environments one would prefer shorter sentences. Then, it is not sufficient to obtain sentences with uniform surprisal profiles, they also have to be close to the capacity of the current communication channel in order to be considered efficient. Therefore, it is important to be able to estimate the capacity of the communication channel, which to these days remains rather unclear.

These issues remain for future work, where the architecture proposed here might serve as starting point towards an account of UID at the algorithmic level of analysis.

Chapter 8

Conclusion

The task of modeling human sentence production was approached from a connectionist point of view and using distributed semantic representations. The results have led to different conclusions in each chapter, which will be summarized here.

This work focused on three main topics: i) the use of distributed representations for connectionist modeling of language production concerning systematicity, ii) the internal dynamics of language production models with recurrent neural networks, and iii) the implementation of a model that reflects the intuitions of UID.

Considering the semantic representations that were used as input for the models (situation and belief vectors), these possess some properties that facilitate the modeling of certain aspects of human language processing, compared to symbolic discreet representations. First, each representation is a multidimensional continuous vector that corresponds to a point in a vector space; consequently, the number of representations that can be obtained from that space is possibly infinite, an attribute that is necessary if one wants to represent infinite sets, such as the set of sentences that are related to a language. Second, these representations contain full descriptions of the situation models that are related to each sentence, allowing for direct inference and for the estimation of probabilistic information related to the events conveyed in each semantic representation. Finally, another importat property of DSS representations is that since they are continuous vectors, similarity between representations can be assessed simply by measuring the distance between the related vectors. This property is important as was argued in Chapter 5 in order for the model to exhibit systematicity. As explained in said chapter, the input representations have to be comparable amongst each other, in order for a model to draw relations from known inputs to novel ones and thus achieve generalization.

Because of these properties, situation and belief vectors can be regarded as good candidates to model human language processing, in this case language production, compared to symbolic discrete representations; something that was reflected in the positive results of the simulations.

Indeed, concerning systematicity, the language production model was able to learn to produce sentences generalizing across novel input representations and with a very good performance in all test conditions. Furthermore, the errors that were elicited were related to messages that were very similar, demonstrating that the model was processing similar representations in a similar way, and thus further demonstrating systematicity. Moreover, these errors matched relevant findings reported in the speech error literature, where speech errors are elicited involving similar elements. Finally, the model was not only able to produce one encoding per message representations, but most if not all of the encodings that the microlanguage allows, thereby further demonstrating systematicity. All of this was shown using an architecture very similar if not identical to the one of the Prod-SRN model of Chang (2002), suggesting that perhaps the level of systematicity needed for human language processing can be achieved without the need for a highly complex architecture as the Dual-Path model (Chang, 2002).

While the behavior of the production model showed that it was able to produce correct sentences exhibiting systematicity, the internal mechanism of the model was still rather unclear and, because of that, an analysis of its internal dynamics was performed. This analysis consisted on tracking the flow of activation that goes from the input units to the output units, showing the relations of activation or inhibition that exist between different units. As we saw, each input pattern promotes the activation of words that are related to its semantics, while inhibiting the words that are in conflict. After the production of each word, syntactic and semantic constraints are introduced that condition the following word productions, while the context units preserve information through time. In this view, the behavior of the model can be explained by a set of units where each unit has a specific function and the model learns the proper interactions between units such that a correct global behavior arises. Then, a correct processing of novel inputs results if the rules learned during training that govern the interactions between units is coherent with the rules of the production task.

The analysis also revealed patterns that have been previously suggested with regards to language processing with recurrent neural networks: some hidden units seem to be related to specific functions (Karpathy et al., 2015), the activation patterns related to each word reflect similarity between words (Mikolov et al., 2013), and the general intuition that the recurrence serves to preserve information over time. The results presented here further demonstrate such intuitions and provide a mechanistic holistic account of how the model works, which could give insight into the mechanism that is implemented by other models with similar architecture.

Concerning an implementation of UID, an extension of the sentence production model was proposed, profiting from the ability of said model to produce several encodings for a given message representation. This extension implements the two main rational goals that have been suggested to govern UID: on the one hand, the model can show a tendency to maximize the amount of information transmitted per time step, being quick; and on the other hand, the model tries to stay below the limit of the channel capacity in order avoid miscommunication errors. By balancing these two tendencies one could achieve different levels of uniformity, providing a starting point to model UID at the algorithmic level of analysis.

During the development of the UID model some issues arose highlighting some aspects that need to be further explored. First, the information processing from the point of view of the comprehender is different from the point of view of the producer: the producer knows beforehand the message that he/she tries to convey, while the comprehender needs to infer the intended message from the sentence produced. Then, the effort of production is related more to the access of the linguistic units needed to build the sentence, while the effort of comprehension would be related more to the recognition of said units and the construction of a message representation coherent with the sentence. Second, considering that the information content related to a given sentence is related both to the probability of the event that the sentence conveys and the probability of the linguistic elements used to build the sentence, in order to properly assess the information content of sentences it is necessary to build a dataset that reflects in a naturalistic fashion the relative distribution of events and their respective sentences, such that a model trained on such dataset is able to infer how likely a given event is, and how likely a given sequence of linguistic units is. Finally, it is necessary to define the upperbound or capacity of the communication channel. A given sentence can have a very uniform surprisal profile while transmitting very little or very much information per time unit. Then, while uniform surprisal profiles are desirable, they have to be uniform with respect to the channel capacity, which to this day is not clear.

While the results in this dissertation offered evidence to draw some conclusions, they also highlighted some areas that could be explored further in future work:

- **Naturalistic dataset:** The dataset used to train the models here was artificially generated. This allowed a very controlled setting, but raises the question of how this framework could work on real-world data. One possibility would be the use of images or videos as semantic representations and turn the task into a caption generation task. Another possibility would be to use some existing datasets of sentence-semantics items, as the one of Modi et al. (2016), with possible combinations of different datasets and unsupervised methods. The latter is in view of the large quantity of training items that are needed to train large-scale neural networks.

- **Systematicity of complex syntax:** This dissertation approached the topic of systematicity through different test conditions related to different degrees of generalization. Further testing in this direction could serve to determine whether the approach pursued here is able to account for systematicity in the test conditions of Chang (2002), where the model is tested with more difficult structures such as sentences of the form ”a blicket is a blicket”, where a word has to appear in two different positions where it has not been placed before (Marcus, 1998b).

- **Systematicity of impossible/imaginary events:** One benefit of using belief vectors over situation vectors is that the former are not linked to prior probabilities, which means that they have the potential to be used to represent events with zero probability. Future work could include the definition or construction of such representations and testing whether the model would be able to produce sentences for them.

- **Production dynamics of larger models:** The methodology of Chapter 6 could be applied to describe the dynamics of larger models trained on real-world data, such as those used in computational linguistics for language modeling, machine translation or caption generation.

- **UID dataset:** As mentioned in Chapter 7, one issue that remained was the testing using a more naturalistic dataset, although still artificial, that could permit the manipulation of differences between semantic and linguistic surprisal, such that the results of the UID modeling are more conclusive.

In general, the results show that indeed the distributed semantic representations of Frank et al. (2009) can be used to model sentence production, exhibiting systematicity and in general a high performance in all the simulations. Additionally, we arrived to an algorithmic account of the behavior of the model by analyzing its internal structure, and which has the potential to generalize to other models with similar architecture. And finally, we saw that UID can be approached using the proposed model, and by doing so, we were able to detect some aspects that need to be defined in order to go from the formulation of UID at the computational level of analysis to a mechanistic account at the algorithmic level.

Appendix A

Sample of Sentences of the Microlanguage

The following is a random sample of 100 of the sentences that were obtained from the microlanguage of Frank et al. (2009). It is intended to given an idea of the kind of syntactic and semantic patterns that are present. A full list of the sentences can be found in the following link:

`https://github.com/iesus/thesis-production-models/blob/master/sentences.txt`

- the boy beats heidi with ease inside .
- someone beats charlie at hide_and_seek with difficulty .
- hide_and_seek is played in the playground .
- someone beats charlie in the playground at a game .
- heidi plays with a toy outside .
- sophia loses to someone in the bedroom at hide_and_seek .
- charlie wins with ease at a game .
- football is won with ease by the boy outside .
- charlie loses in the street at football .
- heidi beats the boy in the bathroom .
- a girl beats someone at a game .
- charlie wins with difficulty at football in the street .
- soccer is lost by charlie in the street .

- a game is won by charlie in the playground .
- a girl loses to sophia at football .
- charlie beats heidi with ease at hide_and_seek in the shower .
- charlie beats a girl in the bedroom .
- a girl beats charlie at football in the street .
- sophia beats someone with ease outside .
- sophia loses to someone outside .
- sophia loses to someone at football outside .
- football is won outside .
- sophia loses to the boy in the playground at hide_and_seek .
- the boy loses to sophia inside at hide_and_seek .
- heidi loses to a girl at a game inside .
- someone loses to a girl at football .
- someone beats the boy at chess with difficulty .
- soccer is played badly by a girl in the street .
- the boy beats a girl with ease at a game outside .
- charlie plays a game badly in the bedroom .
- the boy loses at hide_and_seek in the playground .
- charlie beats sophia in the bathroom .
- sophia wins with difficulty at football in the street .
- football is won with difficulty by sophia .
- a game is played by charlie in the playground .
- football is lost by sophia in the street .
- the boy beats sophia at chess .
- someone beats a girl at soccer outside .
- a toy is played with by the boy .
- charlie beats heidi with difficulty at chess inside .
- a game is played by the boy in the playground .
- sophia loses in the shower at hide_and_seek .
- heidi plays football badly .
- someone beats sophia at a game with ease .

- a game is played in the street .
- a girl loses to someone at a game in the playground .
- a girl wins at football .
- sophia beats a girl with ease at a game outside .
- heidi beats the boy with ease at hide_and_seek in the shower .
- someone loses in the shower at hide_and_seek .
- someone beats sophia in the bathroom at hide_and_seek .
- hide_and_seek is played by heidi in the bathroom .
- a game is lost by the boy in the shower .
- a jigsaw is played with by heidi in the bedroom .
- heidi loses to the boy in the bedroom at chess .
- heidi beats sophia at a game with ease .
- a girl beats charlie with ease at hide_and_seek in the bathroom .
- hide_and_seek is played well by heidi inside .
- sophia wins with difficulty at hide_and_seek in the shower .
- someone loses in the street at soccer .
- football is won with difficulty .
- a girl beats sophia with ease at chess inside .
- sophia plays hide_and_seek well in the shower .
- hide_and_seek is played by a girl .
- heidi beats sophia with ease at hide_and_seek in the bathroom .
- a girl beats charlie with difficulty at a game .
- charlie beats a girl with difficulty at hide_and_seek in the bedroom .
- a girl loses to someone at hide_and_seek in the bedroom .
- charlie loses to heidi in the shower .
- a ball is played with in the playground by a girl .
- charlie wins at soccer outside .
- a doll is played with in the bedroom by sophia .
- soccer is played by charlie in the street .
- heidi beats a girl outside at football .
- the boy beats sophia with difficulty in the bathroom .

- a girl beats charlie in the bathroom .
- heidi beats charlie at a game with difficulty .
- sophia beats someone with ease at soccer in the street .
- someone beats charlie with difficulty at hide_and_seek in the bathroom .
- charlie beats heidi with difficulty at football outside .
- sophia beats the boy with ease outside .
- charlie beats a girl with difficulty at soccer outside .
- hide_and_seek is played by the boy .
- sophia loses to charlie at hide_and_seek .
- someone wins with difficulty in the bathroom .
- a girl beats heidi inside at hide_and_seek .
- the boy loses to sophia in the bathroom at a game .
- hide_and_seek is played well by someone inside .
- sophia beats charlie with ease at soccer in the street .
- football is played by heidi outside .
- a girl beats someone with ease in the shower .
- a girl beats heidi with difficulty at a game outside .
- hide_and_seek is won with ease by a girl .
- sophia loses at chess .
- the boy loses to a girl at football outside .
- hide_and_seek is won with difficulty by a girl in the shower .
- a girl beats sophia at hide_and_seek .
- heidi loses to the boy at a game inside .
- heidi beats sophia at football .
- hide_and_seek is played badly by a girl outside .

Appendix B

Language Production Model Code

The basic structure of the language production model presented in Chapter 4 is defined by the following code, which was implemented using Python 2.7 and the library Theano. This class is later used by the functions shown in Appendix C.

```
import theano,numpy,os
from theano import tensor as T
from collections import OrderedDict


'''
        Recurrent Neural Network to model language production, receives a DSS input and outputs a word per time
        step it learns with backpropagation. One could also use backpropagation through time, however, there is
        no apparent difference in performance, it only takes more time to train if one uses backpropagation
        through time.
'''

class model(object):
        def __init__(self, inputDimens,hiddenDimens,outputDimens):

                def sample_weights(sizeX, sizeY):
                        values = numpy.ndarray([sizeX, sizeY], dtype=theano.config.floatX)
                        for dx in xrange(sizeX):
                                vals=numpy.random.normal(loc=0.0, scale=0.1,size=(sizeY,))
                                values[dx,:] = vals
                        return values

                # parameters of the model
                self.W_xh   = theano.shared(sample_weights(inputDimens,hiddenDimens))
                self.W_oh   = theano.shared(sample_weights(outputDimens,hiddenDimens))
                self.W_hh   = theano.shared(sample_weights(hiddenDimens,hiddenDimens))
                self.W_hy   = theano.shared(sample_weights(hiddenDimens,outputDimens))

                self.bh  = theano.shared(numpy.zeros(hiddenDimens, dtype=theano.config.floatX))
                self.b   = theano.shared(numpy.zeros(outputDimens, dtype=theano.config.floatX))

                #fixed constants
                self.h0  = numpy.zeros(hiddenDimens, dtype=theano.config.floatX)  # @UndefinedVariable
                self.o0  = numpy.zeros(outputDimens, dtype=theano.config.floatX)  # @UndefinedVariable

                # bundle
                self.params = [self.W_xh,self.W_oh, self.W_hh, self.W_hy, self.bh, self.b]
                self.names  = ['W_xh','W_oh','W_hh', 'W_hy', 'bh', 'b']
                dss = T.vector("dss")
                wordLoc    = T.vector("y") #words in localist representation
                h_tm1 = T.vector("h_tm1")
                o_tm1 = T.vector("o_tm1")

                h_t = T.nnet.sigmoid(T.dot(dss, self.W_xh) + T.dot(h_tm1, self.W_hh) + \
                                     T.dot(o_tm1,self.W_oh)+ self.bh)

                outputWordProbs = T.nnet.softmax(T.dot(h_t, self.W_hy) + self.b)
                wordPredLoc = T.argmax(outputWordProbs,axis=1)

                # loss, gradients and learning rate
                lr = T.scalar('lr')
                loss =-T.mean(wordLoc * T.log(outputWordProbs) + (1.- wordLoc) * T.log(1. - outputWordProbs))
                #Cross entropy loss

                gradients = T.grad(loss, self.params )
                updates = OrderedDict(( p, p-lr*g ) for p, g in zip( self.params , gradients))

                # theano functions
                self.classify = theano.function(inputs=[dss,h_tm1,o_tm1],
                                                outputs=[wordPredLoc,h_t,outputWordProbs[0]])
                                                #outputWordProbs at this time is the future o_tm1

                self.train = theano.function( inputs  = [dss, wordLoc, lr,h_tm1,o_tm1],
                                             outputs = [loss,h_t,outputWordProbs[0]],
                                             updates = updates )

        def save(self, folder):
                for param, name in zip(self.params, self.names):
                        numpy.save(os.path.join(folder, name + '.npy'), param.get_value())

        def load(self, folder):
                for param, name in zip(self.params, self.names):
                        values =numpy.load(os.path.join(folder, name + '.npy'))
                        param.set_value(values)

        def epochTrain(self,trainSet,learningRate):
                '''
                Takes the randomized training set (a set of TrainingElement) and trains an epoch
                returns a list with error values for each 25th training item
                '''
                errors=[]
                for sentIndex in xrange(len(trainSet)):
                        sentence=trainSet[sentIndex]

                        h_tm1=self.h0
                        o_tm1=self.o0
                        errSent=0
                        for word in sentence.wordsLocalist:
                                [e,h_tm1,_]=self.train(sentence.input,word,learningRate,h_tm1,o_tm1)
                                o_tm1=word

                                errSent+=e

                if sentIndex%25==0:
                        errors.append(errSent)

                return errors


        def getSentenceProb(self,semantics,wordsLocalist):
                '''
                Takes a semantic representation, and a sentence in localist form, and calculates its
                conditional probability P(sent|DSS)
                '''
                sentenceWordIndices=[numpy.argmax(localist) for localist in wordsLocalist]
                wordInLoc=self.o0
                h_tm1=self.h0
                sentP=1.0

                for wordOutLoc,wordIndex in zip(wordsLocalist,sentenceWordIndices):
                        [_,h_tm1,outProbs]=self.classify(semantics,h_tm1,wordInLoc)
                        wordInLoc=wordOutLoc
                        wordP=outProbs[wordIndex]
                        sentP=sentP*wordP
                        print wordP

                return sentP


        def getModelProductions(self,testSet,periods=True):
                '''
                Takes a testSet (a list of TrainingElement) and whether the predictions should stop by a period or by
                the expected sentence length.
                Returns the word indices of the sentences produced by the model
                '''
                productions=[]

                for item in testSet:
                        sentenceProduced=[]
                        h_tm1=self.h0
                        o_tm1=self.o0
                        predWord=0

                        if periods:
                                while predWord<42 and len(sentenceProduced)<20:
                                    [predWord,h_tm1,o_tm1]=self.classify(item.input,h_tm1,o_tm1)
                                    predWord=predWord[0]
                                    o_tm1=self.o0.copy()
                                    o_tm1[predWord]=1.0

                                    sentenceProduced.append(predWord)
                        else:
                                for _ in xrange(len(item.wordsLocalist)):
                                    [predWord,h_tm1,o_tm1]=self.classify(item.input,h_tm1,o_tm1)
                                    predWord =predWord[0]
                                    o_tm1=self.o0.copy()
                                    o_tm1[predWord]=1.0

                                    sentenceProduced.append(predWord)

                        productions.append(sentenceProduced)
                return productions
```

Appendix C

Training/Testing Code

The following code contains the main methods used to train and test the language production model presented in Chapter 4 and tested in Chapter 5. This code was written using Python 2.7 and the library Theano.

While this code references classes not contained here, it serves to give an idea of the pipeline and hyperparameters that were used to train and test the model. A full working version can be found in:

`https://github.com/iesus/thesis-production-models`

```
import numpy,random, os, sys
import matplotlib.pyplot as plt

import data.loadFiles as loadFiles
from data.crossValidation import Fold
from tools.similarities import levenSimilarity
from tools.plusplus import xplusplus
import rnn.prodSRNN_notBPTT_mon

sys.path.append(”../data”)
corpusFilePath=”../data/dataFiles/files−thesis/trainTest_Cond–thesis_0.pick”
wordLocalistMapPath=’../data/dataFiles/map_localist_words.txt’
outputsPath=”../outputs”


def localistToIndices(localistMatrix):
        return [numpy.argmax(localist) for localist in localistMatrix]

def indicesToWords(indices,indexWordMapping):
        return [indexWordMapping[index] for index in indices]

def wordsToIndices(words,wordIndexMapping):
        return[wordIndexMapping[word] for word in words]

def getEquivSentencesIndicesSet(trainElem):
        return [localistToIndices(equivalent.wordsLocalist) for equivalent in trainElem.equivalents]



def getFolders(outputsPath, params):
        ”””
        Creates the 3 folders where all results/models will be stored
        folderThisRun: folder containing all the files of this particular run, will be contained in
        folderThisModel which contains all runs of this specific python file

        bestModel: parameters that achieved best performance on the training set
        lastModel: parameters that the model has at the end of training
        ”””

        #Create folder that contains all the runs for this python file
        currentFolder=outputsPath+”/”+os.path.basename(__file__).split(’.’)[0]
        folderThisModel=currentFolder+”_outputs”

        if not os.path.exists(folderThisModel): os.mkdir(folderThisModel)

        #Create folder for all the files of this specific run
        folderThisRun=folderThisModel+”/output”

        folderThisRun+=”_”+params[’inputType’]
        folderThisRun+=”_”+str(params[’nhidden’])+”h”
        folderThisRun+=”_”+str(params[’lr’])+”lr”
        folderThisRun+=”_”+str(params[’nepochs’])+”ep”
        if params[’periods’]: folderThisRun+=”_dots”
        folderThisRun+=”_”+params[’label’]

        if not os.path.exists(folderThisRun): os.mkdir(folderThisRun)

        #Create folder for plots
        plotsFolder=folderThisRun+”/plots”
        if not os.path.exists(plotsFolder): os.mkdir(plotsFolder)

        #Create folders for best and last model parameters
        bestModel=folderThisRun+”/bestModel”
        if not os.path.exists(bestModel): os.mkdir(bestModel)
        lastModel=folderThisRun+”/lastModel”
        if not os.path.exists(lastModel): os.mkdir(lastModel)

        return folderThisRun,bestModel,lastModel,plotsFolder



def mostSimilarEquivalentsLevens(trainingElement,modelProduction):
        ’’’
        Compares the sentence produced by the model with the set of possible sentences related to the DSS,
        obtains the most similar one with its similarity score
        ’’’
        #Get the possible sentences using word indices
        equivalentsIndices=[localistToIndices(equivalent.wordsLocalist) for equivalent in trainingElement.equivalents]
        #Compare each possible sentence with the sentence the model produced
        similarities=[levenSimilarity(eq,modelProduction) for eq in equivalentsIndices]
        #Get the most similar one
        mostSimilar=numpy.argmax(similarities, 0)

        return (similarities[mostSimilar],equivalentsIndices[mostSimilar])


def evaluateSRNN(srnn, outFile, evalSet):
        productions_test=srnn.getModelProductions(evalSet)

        simgolds=[mostSimilarEquivalentsLevens(sent,pred) for sent,pred in zip(evalSet,productions_test)]
        similarities=[acc for (acc,_) in simgolds]
        golds=[gold for (_,gold) in simgolds]

        predWords=[indicesToWords(pred,mapIndexWord) for pred in productions_test]
        labelWords=[indicesToWords(label,mapIndexWord) for label in golds]

        printResults(outFile,predWords,labelWords,similarities,evalSet)

def printResults(outFile, predWords,labelWords, similarities, evalSet):
        accuracyGlobal=numpy.sum(similarities)/len(similarities)

        perfect=[]
        almostPerfect=[]
        mildlyBad=[]
        worst=[]

        def printSubSet(label,setValues,superSize):
```

```
        print label
        outFile.write(label+”\n”)

        for (pw,lw,_) in setValues:
            print pw,lw
            outFile.write(str(pw)+” ”+str(lw)+”\n”)

        print len(setValues)  #number of sentences that fell under this range
        outFile.write(str(len(setValues))+”\n”)
        print len(setValues)/float(superSize)#proportion of these sentences with respect to whole condition
        outFile.write(str(len(setValues)/float(superSize))+”\n\n”)
        print

    for pw, lw, acc,item in zip(predWords, labelWords, similarities, evalSet):
        print item.testItem
        print pw,lw
        print acc
        print
        outFile.write(item.testItem+”\n”)
        outFile.write(str(pw)+” ”+str(lw)+”\n”)
        outFile.write(str(acc)+”\n\n”)

        if acc==1.0: perfect.append((pw,lw,acc))
        elif acc>=0.8: almostPerfect.append((pw,lw,acc))
        elif acc>=0.5: mildlyBad.append((pw,lw,acc))
        else: worst.append((pw,lw,acc))

    printSubSet(”PERFECT INSTANCES”,perfect,len(evalSet))
    printSubSet(”ALMOST PERFECT”,almostPerfect,len(evalSet))
    printSubSet(”MILDLY BAD”, mildlyBad,len(evalSet))
    printSubSet(”WORST INSTANCES”, worst,len(evalSet))

    print
    print accuracyGlobal
    outFile.write(”\n”+str(accuracyGlobal)+”\n”)


if __name__ == ’__main__’:

    if len(sys.argv)>1:
    x=1
    s={
        ’lr’:float(sys.argv[xplusplus(”x”)]),          #learning rate
        ’decay’:int(sys.argv[xplusplus(”x”)]),         #decay on the learning rate if improvement stops
        ’nhidden’:int(sys.argv[xplusplus(”x”)]),       #number of hidden units
        ’seed’:int(sys.argv[xplusplus(”x”)]),          #seed for random
        ’nepochs’:int(sys.argv[xplusplus(”x”)]),       #max number of training epochs
        ’label’:sys.argv[xplusplus(”x”)],              #label for this run
        ’periods’:int(sys.argv[xplusplus(”x”)]),       #whether the corpus has periods
        ’load’:int(sys.argv[xplusplus(”x”)]),          #whether the model is already trained or not
        ’inputType’:sys.argv[xplusplus(”x”)],          #dss or sitVector or compVector
        ’actpas’:sys.argv[xplusplus(”x”)],             #if the inputs are divided in actpas
        ’inputFile’:sys.argv[xplusplus(”x”)]           #FILE containing the input data
    }

    else:
    s = {
        ’lr’:0.24,                  #learning rate
        ’decay’:True,               #decay on the learning rate if improvement stops
        ’nhidden’:120,              #number of hidden units
        ’seed’:345,                 #seed for random
        ’nepochs’:200,              #max number of training epochs
        ’label’:”15_40_monitor_sigm_anew1”,     #label for this run
        ’periods’:True,             #whether the corpus has periods
        ’load’:True,                #whether the model is already trained or not
        ’inputType’:’beliefVector’,#dss or sitVector or compVector
        ’actpas’:True,              #if the inputs are divided in actpas
        ’inputFile’:corpusFilePath  #FILE containing the input data
    }
    if s[’periods’]: s[’vocab_size’]=43
    else: s[’vocab_size’]=42

    if s[’inputType’]==’sitVector’ or s[’inputType’]==’compVector’ or s[’inputType’]==”beliefVector”:
        s[’inputDimension’]=44
    if s[’inputType’]==’dss’: s[’inputDimension’]=150
    if s['actpas']:s['inputDimension']=s['inputDimension']+1

    #LOAD FILES
    mapIndexWord=loadFiles.getWordLocalistMap(wordLocalistMapPath)

    fold=Fold()
    fold.loadFromPickle(s['inputFile'])
    trainLists=fold.trainSet
    testLists=fold.valtestSet

    loadFiles.setInputType(trainLists[0],s['inputType'])
    for tList in testLists:
        loadFiles.setInputType(tList,s['inputType'])

    train=trainLists[0]
    validateList=trainLists[1]# Traintest is used instead of validation

    folderThisRun,bestModel,lastModel,plotsFolder=getFolders(outputsPath,s)

    #CREATE SRNN AND INITIALIZE VARS
    srnn = rnn.prodSRNN_notBPTT_mon.model(
                                  inputDimens=s['inputDimension'],
                                  hiddenDimens = s['nhidden'],
                                  outputDimens= s['vocab_size']
                                  )
    random.seed(s['seed'])

    #IF THE MODEL HASNT BEEN TRAINED
    if not s['load']:
        outputFile= open(folderThisRun+'/output.txt', 'w+')
        best_sim = -numpy.inf
        bestEp=0
        epErrors=[]
        epSimilarities=[]
        s['clr'] = s['lr']

        for epoch in xrange(s['nepochs']):
                random.shuffle(train)

                #TRAIN THIS EPOCH
                errors=srnn.epochTrain(train,s['clr'])
                epErrors.append(sum(errors))

                predictions_validate=srnn.getModelProductions(validateList,False)#We don't stop on
                #periods because at the beginning the model may not know that it has to put a period

                #Get a list of pairs (sim,mostSimilar) where sim is the similarity of the most similar
                #sentence in the gold sentences of the given dss
                simgolds=[mostSimilarEquivalentsLevens(sent,pred) for sent,pred in \
                                              zip(validateList,predictions_validate)]

                #Get only the list of similarities
                similarities=[sim for (sim,mostSimilar) in simgolds]
                similarity=numpy.sum(similarities)/len(similarities)
                epSimilarities.append(similarity)

                outputLine='Epoch: '+str(epoch)+' lr: '+str(s['clr'])+' similarity: '+str(similarity)

                if similarity > best_sim:
                        srnn.save(bestModel)
                        best_sim = similarity
                        bestEp=epoch
                        lastChange_LR=epoch
                        outputLine='NEW BEST '+outputLine

                outputFile.write(outputLine+'\n')
                print outputLine

                errorsPlot=plt.figure(100000)
                plt.plot(epErrors)
                plt.savefig(folderThisRun+"/errorsTrainEp.png")

                simPlot=plt.figure(1000000)
                plt.plot(epSimilarities)
                plt.savefig(folderThisRun+"/similarities.png")
```

```
            # learning rate halves if no improvement in 15 epochs
            if s['decay'] and (epoch-lastChange_LR) >= 15:
                s['clr'] *= 0.5
                lastChange_LR=epoch#we have to reset lastChange_LR,
                #otherwise it will halve each epoch until we get an improvement

            #TRAINING STOPS IF LEARNING RATE IS BELOW THRESHOLD OR IF NO IMPROVEMENT IN 40 EPOCHS
            if s['clr'] < 1e-3 or (epoch-bestEp)>=40:
                break

        srnn.save(lastModel)
        outputLine='BEST RESULT: epoch '+str(bestEp)+' Similarity: '+str(best_sim)+ \
                                ' with the model '+folderThisRun
        print outputLine
        outputFile.write(outputLine)
        outputFile.close()

    else:
        #IF THE MODEL WAS ALREADY TRAINED AND WE ARE ONLY LOADING IT FOR TESTING
        srnn.load(lastModel)

        outFileTrain= open(folderThisRun+'/outputlast_train.txt', 'w+')
        outFileTest= open(folderThisRun+'/outputlast_test.txt', 'w+')

        evaluateSRNN(srnn, outFileTrain, validateList)
        outFileTrain.close()

        for index in xrange(len(testList)):
            print "\nCONDITION:"+str(index+1)+"\n"
            outFileTest.write("\nCONDITION:"+str(index+1)+"\n")
            evaluateSRNN(srnn, outFileTest, testLists[index])

        outFileTest.close()
```

Bibliography

John Anderson. The place of cognitive architectures in rational analysis. In K. VanLehn, editor, *Architectures for Cognition*. Lawrence Erlbaum, Hillsdale, NJ, 1991.

Jennifer E Arnold, Anthony Losongco, Thomas Wasow, and Ryan Ginstrom. Heaviness vs. newness: The effects of structural complexity and discourse status on constituent ordering. *Language*, 76(1):28–55, 2000.

Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. Explaining recurrent neural network predictions in sentiment analysis. *arXiv preprint arXiv:1706.07206*, 2017.

Matthew Aylett and Alice Turk. The smooth signal redundancy hypothesis: A functional explanation for relationships between redundancy, prosodic prominence, and duration in spontaneous speech. *Language and speech*, 47(1):31–56, 2004.

Matthew Aylett and Alice Turk. Language redundancy predicts syllabic duration and the spectral characteristics of vocalic syllable nuclei. *The Journal of the Acoustical Society of America*, 119(5):3048–3058, 2006.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

Lawrence W Barsalou. Perceptions of perceptual symbols. *Behavioral and brain sciences*, 22(4):637–660, 1999.

William Bechtel. The case for connectionism. *Philosophical studies*, 71(2):119–154, 1993.

Alan Bell, Daniel Jurafsky, Eric Fosler-Lussier, Cynthia Girand, Michelle Gregory, and Daniel Gildea. Effects of disfluencies, predictability, and utterance position on word form variation in english conversation. *The Journal of the Acoustical Society of America*, 113(2):1001–1024, 2003.

Alan Bell, Jason M Brenier, Michelle Gregory, Cynthia Girand, and Dan Jurafsky. Predictability effects on durations of content and function words in conversational english. *Journal of Memory and Language*, 60(1):92–111, 2009.

Kathryn Bock. Meaning, sound, and syntax: Lexical priming in sentence production. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 12(4):575, 1986a.

Kathryn Bock. Syntactic persistence in language production. *Cognitive psychology*, 18 (3):355–387, 1986b.

Kathryn Bock. An effect of the accessibility of word forms on sentence structures. *Journal of memory and language*, 26(2):119–137, 1987.

Kathryn Bock. Closed-class immanence in sentence production. *Cognition*, 31(2): 163–186, 1989.

Kathryn Bock and Zenzi M Griffin. The persistence of structural priming: Transient activation or implicit learning? *Journal of experimental psychology: General*, 129(2): 177, 2000.

Kathryn Bock and Helga Loebell. Framing sentences. *Cognition*, 35(1):1–39, 1990.

Kathryn Bock and Richard K Warren. Conceptual accessibility and syntactic structure in sentence formulation. *Cognition*, 21(1):47–67, 1985.

Kathryn Bock, Gary S Dell, Zenzi Griffin, Franklin Chang, and Victor S Ferreira. Structural priming as implicit learning. In *meeting of the Psychonomic Society, Chicago, IL*, 1996.

Mikael Bodén. Generalization by symbolic abstraction in cascaded recurrent networks. *Neurocomputing*, 57:87–104, 2004.

Philémon Brakel and Stefan L Frank. Strong systematicity in sentence processing by simple recurrent networks. In *31th Annual Conference of the Cognitive Science Society (COGSCI-2009)*, pages 1599–1604. Cognitive Science Society, 2009.

Harm Brouwer. *The Electrophysiology of Language Comprehension: A Neurocomputational Model*. PhD thesis, University of Groningen, 2014.

Harm Brouwer, Matthew W Crocker, Noortje J Venhuizen, and John CJ Hoeks. A neurocomputational model of the n400 and the p600 in language processing. *Cognitive science*, 41:1318–1352, 2017.

Raluca Budiu and John R Anderson. Interpretation-based processing: A unified theory of semantic sentence comprehension. *Cognitive Science*, 28(1):1–44, 2004.

Brian Butterworth. Some constraints on models of language production. *Language Production: Vol. 1. Speech and Talk*, 1980.

Brian Butterworth. Speech errors: Old data in search of new theories. *Linguistics*, 19 (7-8):627–662, 1981.

Jesús Calvillo. Fast and easy: Approximating uniform information density in language production. In *Proceedings of the 39th Annual Conference of the Cognitive Science Society.* Austin, TX: Cognitive Science Society, 2017.

Jesús Calvillo and Matthew Crocker. Language production dynamics with recurrent neural networks. In *Proceedings of the Eight Workshop on Cognitive Aspects of Computational Language Learning and Processing*, pages 17–26, 2018.

Jesús Calvillo, Harm Brouwer, and Matthew W Crocker. Connectionist semantic systematicity in language production. In *Proceedings of the 38th Annual Conference of the Cognitive Science Society.* Austin, TX: Cognitive Science Society, 2016.

David Carroll. *Psychology of language.* Nelson Education, 2007.

Franklin Chang. Symbolically speaking: A connectionist model of sentence production. *Cognitive science*, 26(5):609–651, 2002.

Franklin Chang. Learning to order words: A connectionist model of heavy np shift and accessibility effects in japanese and english. *Journal of Memory and Language*, 61(3): 374–397, 2009.

Franklin Chang and Hartmut Fitz. Computational models of sentence production: a dual-path approach. *The Oxford handbook of language production*, pages 70–87, 2014.

Franklin Chang, Zenzi M Griffin, Gary S Dell, and Kathryn Bock. Modeling structural priming as implicit learning. *Computational Psycholinguistics, Berkeley, CA*, 29: 392–417, 1997.

Franklin Chang, Gary S Dell, and Kathryn Bock. Becoming syntactic. *Psychological review*, 113(2):234, 2006.

Xinlei Chen and C Lawrence Zitnick. Mind's eye: A recurrent visual representation for image caption generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2422–2431, 2015.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL `http://arxiv.org/abs/1406.1078`.

Herbert H Clark and Eve V Clark. *Psychology and language. An introduction to psycholinguistics.* Harcourt College Pub, 1980.

Alexandra A Cleland and Martin J Pickering. The use of lexical and syntactic information in language production: Evidence from the priming of noun-phrase structure. *Journal of Memory and Language*, 49(2):214–230, 2003.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Judith Degen and T Florian Jaeger. Speakers sacrifice some (of the) precision in conveyed meaning to accommodate robust communication. In *Proceedings of the 85th annual meeting of the Linguistic Society of America*, 2011.

Gary S Dell. A spreading-activation theory of retrieval in sentence production. *Psychological review*, 93(3):283, 1986.

Gary S Dell and Padraig G O'seaghdha. Mediated and convergent lexical priming in language production: A comment on levelt et al (1991). 1991.

Gary S Dell, Cornell Juliano, and Anita Govindjee. Structure and content in language production: A theory of frame constraints in phonological speech errors. *Cognitive Science*, 17(2):149–195, 1993.

Gary S Dell, Myrna F Schwartz, Nadine Martin, Eleanor M Saffran, and Deborah A Gagnon. Lexical access in aphasic and nonaphasic speakers. *Psychological review*, 104 (4):801, 1997.

Gary S Dell, Franklin Chang, and Zenzi M Griffin. Connectionist models of language production: Lexical access and grammatical encoding. *Cognitive Science*, 23(4):517–542, 1999.

Rutvik Desai. A model of frame and verb compliance in language acquisition. *Neurocomputing*, 70(13-15):2273–2287, 2007.

Yanzhuo Ding, Yang Liu, Huanbo Luan, and Maosong Sun. Visualizing and understanding neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1150–1159, 2017.

Andrew W Ellis. On the freudian theory of speech errors. *Errors in linguistic performance: Slips of the tongue, ear, pen, and hand*, pages 123–131, 1980.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Jeffrey L Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, 7(2-3):195–225, 1991.

Jeffrey L Elman. Language as a dynamical system. *Mind as motion: Explorations in the dynamics of cognition*, pages 195–225, 1995.

Jeffrey L Elman. 5 systematicity in the lexicon: On having your cake and eating it too. *The Architecture of Cognition: Rethinking Fodor and Pylyshyn's Systematicity Challenge*, page 115, 2014.

Igor Farkaš and Matthew W Crocker. Syntactic systematicity in sentence processing with a recurrent self-organizing network. *Neurocomputing*, 71(7):1172–1179, 2008.

David Fay and Anne Cutler. Malapropisms and the structure of the mental lexicon. *Linguistic inquiry*, 8(3):505–520, 1977.

Fernanda Ferreira. Choice of passive voice is affected by verb type and animacy. *Journal of Memory and Language*, 33(6):715–736, 1994.

Victor S Ferreira and Gary S Dell. Effect of ambiguity and lexical availability on syntactic and lexical production. *Cognitive psychology*, 40(4):296–340, 2000.

Hartmut Fitz. *Neural syntax*. PhD thesis, University of Amsterdam, 2009.

Jerry A Fodor. *The language of thought*, volume 5. Harvard University Press, 1975.

Jerry A Fodor and Brian P McLaughlin. Connectionism and the problem of systematicity: Why smolensky's solution doesn't work. *Cognition*, 35(2):183–204, 1990.

Jerry A Fodor and Zenon W Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988.

Paul Fraisse. Latency of different verbal responses to the same stimulus. *Quarterly Journal of Experimental Psychology*, 19(4):353–355, 1967.

Austin F Frank and T Florian Jaeger. Speaking rationally: Uniform information density as an optimal strategy for language production. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 30, 2008.

Stefan L Frank and Willem FG Haselager. Robust semantic systematicity and distributed representations in a connectionist model of sentence comprehension. 2006.

Stefan L Frank and Gabriella Vigliocco. Sentence comprehension as mental simulation: an information-theoretic perspective. *Information*, 2(4):672–696, 2011.

Stefan L Frank, Mathieu Koppen, Leo GM Noordman, and Wietske Vonk. Modeling knowledge-based inferences in story comprehension. *Cognitive Science*, 27(6):875–910, 2003.

Stefan L Frank, Willem FG Haselager, and Iris van Rooij. Connectionist semantic systematicity. *Cognition*, 110(3):358–379, 2009.

Sigmund Freud. *The psychopathology of everyday life*, chapter Slips of the tongue. Penguin Books, 1924.

Victoria A Fromkin. The non-anomalous nature of anomalous utterances. *Language*, pages 27–52, 1971.

Victoria A Fromkin. Speech errors as linguistic evidence, the hague: Mouton. 1980. *Errors in Linguistic Performance: Slips of the Tongue, Ear, Pen and Hand. New York*, 1973.

Alan Garnham, Richard C Shillcock, Gordon DA Brown, Andrew ID Mill, and Anne Cutler. Slips of the tongue in the london-lund corpus of spontaneous conversation. *Linguistics*, 19(7-8):805–818, 1981.

Merrill F Garrett. The analysis of sentence production1. In *Psychology of learning and motivation*, volume 9, pages 133–177. Elsevier, 1975.

Merrill F Garrett. Levels of processing in sentence production. *Language production I*, pages 177–220, 1980.

Tim Gelder. Compositionality: A connectionist variation on a classical theme. *Cognitive Science*, 14(3):355–384, 1990.

Dmitriy Genzel and Eugene Charniak. Entropy rate constancy in text. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 199–206. Association for Computational Linguistics, 2002.

Melvyn A Goodale and A David Milner. Separate visual pathways for perception and action. *Trends in neurosciences*, 15(1):20–25, 1992.

Jean K Gordon and Gary S Dell. Learning to divide the labor: An account of deficits in light and heavy verb production. *Cognitive Science*, 27(1):1–40, 2003.

William W Graves, Thomas J Grabowski, Sonya Mehta, and Jean K Gordon. A neural signature of phonological access: distinguishing the effects of word frequency from familiarity and length in overt picture naming. *Journal of cognitive neuroscience*, 19 (4):617–631, 2007.

Robert F Hadley. Systematicity in connectionist language learning. *Mind & Language*, 9(3):247–272, 1994a.

Robert F Hadley. Systematicity revisited: reply to christiansen and chater and niklasson and van gelder. *Mind & Language*, 9(4):431–444, 1994b.

Robert F Hadley and Vlad C Cardei. Language acquisition from sparse input without error feedback. *Neural Networks*, 12(2):217–235, 1999.

Robert F Hadley and Michael B Hayward. Strong semantic systematicity from hebbian connectionist learning. *Minds and Machines*, 7(1):1–37, 1997.

John Hale. A probabilistic earley parser as a psycholinguistic model. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies*, NAACL '01, pages 1–8, Stroudsburg, PA, USA, 2001a. Association for Computational Linguistics.

John Hale. A probabilistic earley parser as a psycholinguistic model. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics, 2001b.

Trevor A Harley. A critique of top-down independent levels models of speech production: Evidence from non-plan-internal speech errors. *Cognitive Science*, 8(3):191–219, 1984.

Robert J Hartsuiker, Sarah Bernolet, Sofie Schoonbaert, Sara Speybroeck, and Dieter Vanderelst. Syntactic priming persists while the lexical boost decays: Evidence from written and spoken dialogue. *Journal of Memory and Language*, 58(2):214–238, 2008.

John Haugeland. An overview of the frame problem. 1987.

John A Hawkins. *A performance theory of order and constituency*, volume 73. Cambridge University Press, 1994.

John A Hawkins. *Efficiency and complexity in grammars*. Oxford University Press on Demand, 2004.

Archibald A Hill. A theory of speech errors. *Speech errors as linguistic evidence*, pages 205–214, 1973.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Steffen Hölldobler, Yvonne Kalinke, and Helko Lehmann. Designing a counter: Another case study of dynamics and activation landscapes in recurrent networks. In *KI-97: Advances in Artificial Intelligence*, pages 313–324. Springer, 1997.

T Florian Jaeger. *Redundancy and syntactic reduction in spontaneous speech.* PhD thesis, Stanford University, 2006.

T Florian Jaeger. Redundancy and reduction: Speakers manage syntactic information density. *Cognitive psychology*, 61(1):23–62, 2010.

Peter A Jansen and Scott Watter. Strong systematicity through sensorimotor conceptual grounding: an unsupervised, developmental approach to connectionist sentence processing. *Connection Science*, 24(1):25–55, 2012.

Jörg D Jescheniak and Willem JM Levelt. Word frequency effects in speech production: Retrieval of syntactic information and of phonological form. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 20(4):824, 1994.

Philip Nicholas Johnson-Laird. *Mental models: Towards a cognitive science of language, inference, and consciousness*, volume 6. Harvard University Press, 1983.

Michael I Jordan. Serial order: a parallel distributed approach (ics report 8604). san diego: University of california. *Institute for Cognitive science*, 1986.

Daniel Jurafsky. A probabilistic model of lexical and syntactic access and disambiguation. *Cognitive Science*, 20(2):137–194, 1996.

Akos Kádár, Grzegorz Chrupała, and Afra Alishahi. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics*, 43(4):761–780, 2017.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078, 2015. URL `http://arxiv.org/abs/1506.02078`.

Roel Kerkhofs and Willem FG Haselager. The embodiment of meaning. *Manuscrito*, 29 (2):753–764, 2006.

Brett Kessler and Rebecca Treiman. Syllable structure and the distribution of phonemes in english syllables. *Journal of Memory and language*, 37(3):295–311, 1997.

Walter Kintsch and Teun A Van Dijk. Toward a model of text comprehension and production. *Psychological review*, 85(5):363, 1978.

Teuvo Kohonen. Self-organizing maps. *Berlin:Springer*, 1995.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553): 436, 2015.

Willem JM Levelt. *Speaking: From intention to articulation*, volume 1. MIT press, 1989.

Willem JM Levelt. Models of word production. *Trends in cognitive sciences*, 3(6): 223–232, 1999.

Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

Roger Levy. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177, 2008.

Roger Levy and T Florian Jaeger. Speakers optimize information density through syntactic reduction. *Advances in neural information processing systems*, 19:849, 2007.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*, 2015.

Gary F Marcus. Can connectionism save constructivism? *Cognition*, 66(2):153–182, 1998a.

Gary F Marcus. Rethinking eliminative connectionism. *Cognitive psychology*, 37(3): 243–282, 1998b.

David Marr. Vision: a computational investigation into the human representation and processing of visual information. w. h. *WH San Francisco: Freeman and Company*, 1982.

Marshall R Mayberry, Matthew W Crocker, and Pia Knoeferle. Learning to attend: A connectionist model of situated language comprehension. *Cognitive science*, 33(3): 449–496, 2009.

Janet L McDonald, Kathryn Bock, and Michael H Kelly. Word and world order: Semantic, phonological, and metrical determinants of serial position. *Cognitive Psychology*, 25(2):188–230, 1993.

Rudolf Meringer and Karl Mayer. *Versprechen und Verlesen. Eine psychologisch-linguistische Studie.([With the assistance of] Carl Mayer.) New edition with an introductory article by Anne Cutler and David Fay*, volume 2. John Benjamins Publishing, 1895.

Risto Miikkulainen. Subsymbolic case-role analysis of sentences with embedded clauses. *Cognitive Science*, 20(1):47–73, 1996.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013.

George Armitage Miller. *The science of words.* Scientific American Library, 1991.

Michele Miozzo and Alfonso Caramazza. Retrieval of lexical–syntactic features in tip-of-the tongue states. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 23(6):1410, 1997.

Mortimer Mishkin and Leslie G Ungerleider. Contribution of striate inputs to the visuospatial functions of parieto-preoccipital cortex in monkeys. *Behavioural brain research*, 6(1):57–77, 1982.

Ashutosh Modi, Tatjana Anikina, Simon Ostermann, and Manfred Pinkal. Inscript: Narrative texts annotated with script information. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may 2016. European Language Resources Association (ELRA). ISBN 978-2-9517408-9-1.

Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *arXiv preprint arXiv:1706.07979*, 2017.

Charles Sanders Peirce. Logic as semiotic: The theory of signs. In Robert E Innis, editor, *Semiotics: An introductory anthology*, pages 4–23. Indiana University Press, 1903/1985.

Steven T Piantadosi, Harry Tily, and Edward Gibson. Word lengths are optimized for efficient communication. *Proceedings of the National Academy of Sciences*, 108(9): 3526–3529, 2011.

Martin J Pickering and Holly P Branigan. The representation of verbs: Evidence from syntactic priming in language production. *Journal of Memory and language*, 39(4): 633–651, 1998.

Martin J Pickering and Victor S Ferreira. Structural priming: a critical review. *Psychological bulletin*, 134(3):427, 2008.

Milena Rabovsky, Steven S Hansen, and James L McClelland. N400 amplitudes reflect change in a probabilistic representation of meaning: Evidence from a connectionist model. In *Proceedings of the 38th Annual Conference of the Cognitive Science Society. Austin, TX: Cognitive Science Society*, 2016.

John Robert Ross. Constraints on variables in syntax. 1967.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948. URL `http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf`.

Stefanie Shattuck-Hufnagel. Speech errors as evidence for a serial order mechanism in sentence production. In *Sentence processing: Psycholinguistic studies presented to Merril Garrett*. Lawrence Erlbaum, 1979.

Hava T Siegelmann. *Neural networks and analog computation: beyond the Turing limit*. Springer Science & Business Media, 2012.

Hava T Siegelmann and Eduardo D Sontag. On the computational power of neural nets. *Journal of computer and system sciences*, 50(1):132–150, 1995.

Mark F St John and James L McClelland. Learning and applying contextual constraints in sentence comprehension. *Artificial intelligence*, 46(1-2):217–257, 1990.

Robert A Stanfield and Rolf A Zwaan. The effect of implied orientation derived from verbal context on picture recognition. *Psychological science*, 12(2):153–156, 2001.

Mark Steijvers and Peter Grünwald. A recurrent network that performs a context-sensitive prediction task. In *Proceedings of the 18th Annual Conference of the Cognitive Science Society*, pages 335–339, 1996.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

John Symons and Paco Calvo. *The architecture of cognition: Rethinking Fodor and Pylyshyn's systematicity challenge*, chapter Systematicity: an overview, pages 3–30. MIT Press, 2014.

Sebastiano Timpanaro. The freudian slip. *New Left Review*, 1(91):43, 1975.

Rob JJH Van Son and Jan PH Van Santen. Duration and spectral balance of intervocalic consonants: A case for efficient communication. *Speech Communication*, 47(1-2): 100–123, 2005.

Noortje J Venhuizen, Matthew W Crocker, and Harm Brouwer. Expectation-based comprehension: Modeling the interaction of world knowledge and linguistic experience. 2018.

Gabriella Vigliocco, Tiziana Antonini, and Merrill F Garrett. Grammatical gender is on the tip of italian tongues. *Psychological science*, 8(4):314–317, 1997.

Hiroko Yamashita and Franklin Chang. “long before short” preference in the production of a head-final language. *Cognition*, 81(2):B45–B55, 2001.

Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

George Kingsley Zipf. Relative frequency as a determinant of phonetic change. *Harvard studies in classical philology*, 40:1–95, 1929.

George Kingsley Zipf. Human behaviour and the principle of least-effort. cambridge ma edn. *Reading: Addison-Wesley*, 1949.

Rolf A Zwaan and Gabriel A Radvansky. Situation models in language comprehension and memory. *Psychological bulletin*, 123(2):162–185, 1998.

Rolf A Zwaan, Robert A Stanfield, and Richard H Yaxley. Language comprehenders mentally represent the shapes of objects. *Psychological science*, 13(2):168–171, 2002.