# GRADED DECOMPOSITIONAL SEMANTIC PREDICTION

by

Adam R. Teichert

A dissertation submitted to Johns Hopkins University in conformity with
the requirements for the degree of Doctor of Philosophy

Baltimore, Maryland
July 2022

# Abstract

Compared to traditional approaches, decompositional semantic labeling (DSL) is compelling but introduces complexities for data collection, quality assessment, and modeling. To shed light on these issues and lower barriers to the adoption of DSL or related approaches I bring existing models and novel variations into a shared, familiar framework, facilitating empirical investigation.

**Primary Reader and Advisor:** Benjamin Van Durme

**Reader:** Mark Dredze

**Reader:** Aaron Steven White

# Acknowledgments

I express deep appreciation to all who have made anything good in this dissertation possible: my Creator, my wife and family, influential teachers (too many to list), my advisors (Quinn Snell, Hal Daumé III, Jason Eisner, and Benjamin Van Durme), other supervisors (Eric Ringger, Scott DuVall, Clair Voss, Adam Lopez, Mark Dredze, Matthew Gormley), all of CLSP, especially Blab, Argo, Agora, and other co-authors including Jonathan Krein, Jiarong Jiang, Tim Vieira, Stephen Mayew, Adrian Benton, Adam Poliak, Rachel Rudinger, Ryan Culkin, and Sheng Zhang, and my colleagues at Snow College.

Although any mistakes in this work are my own, I heartily thank my thesis committee for their patience and valuable feedback.

I thank D. Reisinger for assistance with the SPRL data in Chapter 3 and Tim O'Gorman for his help in that chapter with PropBank and for providing function tag labels for the SPRL annotated sentences that are not in Ontonotes 5. I also appreciate helpful discussion and feedback from anonymous reviewers for various papers.

Much of this research was supported by the Human Language Technology Center of Excellence (HLTCOE) and DARPA LORELEI. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions

contained herein are those of the author and should not be interpreted as representing official policies or endorsements of DARPA or the U.S. Government.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Language provides evidence regarding the existence and nature of real-world or hypothetical entities and events. Semantic prediction seeks to mimic human inferences in order to exploit such evidence. To illustrate, consider the events and entities brought to mind by the following sentences:

(E1) I am writing a dissertation.
(E2) The officer led the suspect to the car.
(E3) The footprints led the officer to the car.
(E4) ‘Twas brillig, and the slithy toves did gyre and gimble in the wabe.

Each sentence evokes some image of participants in some number of events. For example, (E1) describes a *writing* event involving me, the writer, and the dissertation that I am writing.

Language can also evoke fictitious events and entities such as those invented in (E2) and (E3): an officer, a car, a suspect, and even an implied crime. In fact, events and entities implied by language may be entirely impossible or even partially incomprehensible as in (E4). Still, humans are able to harvest inferences about these objects from textual language.

For example, what events are occurring in (E2)? What participants are involved? What role do these participants play in the situation? And, what other attributes of these participants does the sentence reveal or suggest? Indeed several entities, events and attributes can be inferred from the sentence without explicit mention.

Since semantic prediction attempts to mimic human inference from text, *supervised* semantic prediction requires formulating this goal in a way that simultaneously supports:

1. data collection (often human annotation)
2. computational modeling (including a method of inference under the model and a method for optimizing model parameters)
3. evaluation

*Decompositional semantic labeling* has been proposed as a compelling alternative to traditional formalisms, yet practical issues regarding data collection, modeling, and evaluation remain. This thesis sheds light on these issues in order to lower barriers to application of the decompositional approach.

## 1.1 The Case for Decompositional Semantics

Because of the central role of decompositional semantics to the work in this thesis, I include a brief introduction to the topic here.

Traditional natural language processing focused on a small set of core tasks and domains, using detailed annotation instructions and training to allow long-term, trusted annotators to provide supervision for machine learning. However, contemporary trends suggest increasingly many specialized tasks and domains, using more complicated models, and requiring more training data. Consequently, crowd-sourced annotation from large, diverse, and transient pools of lay workers has arisen as a source for the required data. These trends highlight interesting opportunities and problems for tasks that require annotators to lean heavily on unique personal experience (c.f. Aroyo (2013)).

### 1.1.1 Semantic Role Labeling

To mimic human textual inference, we need to formalize the task to be well-suited for human annotation. Chapter 2 explores available options in more depth, but one popular, partial approach is semantic role labeling (SRL). Given an identified predicate “led” in (E2) and an argument “officer”, the task of semantic role labeling has traditionally included (1) selecting the most appropriate “word sense” of the predicate (from a small inventory of word-senses) and (2) selecting a “semantic role” of the argument with respect to that predicate (again, from a small inventory of options). For example, the predicate “led” in (E2) has the sense “directed motion or being ahead of”, the argument “the officer” has the role “agent” (the one doing the leading), “the suspect” has the role “patient” (the one being led), and “the car” has the role “goal” (i.e. the destination of the leading). But although there are inventories that enumerate several word senses and semantic role labels, it is unclear how fine-grained the senses and semantic role categories should be. Furthermore, the practical impact of changing the sense or role inventory after already annotating a large collection of data under an earlier inventory can be significant.

#### 1.1.1.0.1 Early Work

*Semantic Role Labeling* can be traced at least as far back as Gruber (1965) who proposed a few thematic roles such as Theme, Agent, Source, and Goal that are reused in the context of multiple verbs. Early work by Fillmore (1967) also describes a set of covert *deep cases* including *Agentive*, *Instrumental*, *Dative*, *Factive*, *Locative*, *Objective*, and *Benefactive* which he argued exist in the "deep structure" of language and need not be ultimately expressed with morphohological affixes.[1] He cites Whorf (1965) as introducing the idea of "covert categories", and Whorf (1945) highlights out covert categories like "intransitive" and "gender" which have semantic distinction and have syntactic impact.

#### 1.1.1.0.2 FrameNet

Eventually, Fillmore's work evolved into a notion of *Case Frames* where the full meaning of a situation is represented by an entire set of participants filling various roles. This theory was made concrete in an influential data resource known as FrameNet (Baker, Fillmore, and Lowe, 1998). At the time of this writing, FrameNet includes descriptions of 1224 frames, each evoked by potentially many verbs or nouns.[2]

#### 1.1.1.0.3 PropBank

Another popular incarnation of semantic role labeling was operationalized in PropBank (Palmer, Gildea, and Kingsbury, 2005). The major distinction from FrameNet is a move toward sense-specific role slots. In PropBank SRL, an inventory of verb senses are identified

[1] Chomsky argued that syntax could be explained generatively—as the transformation of a *deep* structure of language-specific lexical units connected by language-independent rules and language-specific parameters (Chomsky, 1965).

[2] https://framenet.icsi.berkeley.edu/fndrupal/current_status

and then arguments are labeled with numbered roles that are specific to a particular verb-sense.

### 1.1.2 Proto-Roles

Meanwhile, Dowty (1991) highlighted the difficulty of enumerating a comprehensive inventory of semantic roles and a connection between the *syntactic* property of whether an argument is realized as subject or object and the presence of binary *semantic* properties like awareness or volition.

Dowty suggested several binary properties and categorized them as being prototypical either of agents (the things doing the acting) or of patients (the things being acted on). Returning to our example (E2), proto-*agent* properties of the officer would include volition, sentience, causes-change-of-state, movement, and independent existence, while it is less clear whether or not the proto-*patient* properties (changes-state, incremental theme, causally affected, stationary, and no independent existence) apply.

Such a decompositional approach into overlapping labels is extremely flexible and lends itself particularly well to crowd-sourced annotation. Reisinger et al. (2015) carried out a large-scale empirical investigation of Dowty's theories, using crowdsourcing to acquire property ratings on a Likert-like ordinal scale. For example, rather than labeling whether the suspect was "volitional" as he was led by the officer, annotators provide a judgment on an ordinal scale from "very unlikely" to "very likely".

In principle, guidelines for annotating each property can be described and carried out in isolation from other properties, thereby allowing different questions to be answered in parallel (or even at entirely different times) by different annotators. Annotators, who have no guarantee of the amount of annotation work that will be available for them, are also able to specialize in a subset of properties, further reducing annotator investment. Researchers need not revise previous annotations when new properties of interest are identified. Indeed, previous annotations can even be used to filter annotated data to a more relevant subset for annotation. For example, if we want to identify instances where the sentiment of one argument toward another changes during a predicate, we would only need to annotate those examples where the first argument was already known to be sentient *and* aware of the second argument.

*Graded* annotation also implicitly simplifies annotation guidelines by sidestepping the need to clearly delineate the boundary between positive and negative—a boundary on which consensus may be difficult to achieve. Rather than attempting annotator agreement by iterative training sessions leading to extensive annotation guides, annotators use the graded scale freely to best express their own assessments without needing to study boundary-case instructions or coordinate with other annotators. This approach potentially enables better training and evaluation signal with minimal annotation burden.

Additionally, the promise of lighter annotation investment potentially provides access to a much larger pool of annotators who are unwilling to risk more involved onboarding.

Finally, in addition to theoretic motivations and annotational convenience, the decompositional model exposes unique opportunities for modeling and prediction. Joint probabilistic models of all properties would naturally allow inference of unannotated properties conditioned on available knowledge about the presence or absence of other properties.

## 1.2 Opportunities and Challenges

In summary, a graded, decompositional approach to semantics:

- Supports evaluation of linguistic theories of the syntax-semantics interface.
- Simplifies annotation instructions.
- Allows annotation specialization and parallelization.
- Avoids apriori assumptions of category or question inventory.
- Relaxes the demands on annotator agreement.
- Captures fine-grained, multi-dimensional distinctions in semantic usage.
- Supports joint probabilistic models capable of prediction conditioned on some property observation or prior knowledge.

The overarching goal of this thesis is to elucidate how to make the most of graded, decompositional annotations to build predictive models.

Specifically, I investigate the following questions:

- How can we compare the quality of alternative SPRL systems? (Ch. 3,4)
- How important are inter-property relationships for joint or conditional prediction? (Ch. 4)
- How does a fine-grained (non-binary) training signal impact the quality of a decomp model? (Ch. 4)

- How can an ordinal inductive bias be incorporated into a decomp CRF? (Ch. 5)
- Does an ordinal inductive bias improve a decomp model? (Ch. 5)
- How does approximate inference impact the backprop-based learning of a CRF? (Ch. 6)

## 1.3 Roadmap

The emphasis of the present work is to provide direction for those who want to incorporate graded signal into decompositional semantic prediction models. I formalize a variety of modeling approaches within a unifying family of log-linear conditional random fields (CRFs) amenable to additional log-non-linear variation.

In Chapter 2 I briefly contrast a number of alternative semantic formalisms to the SRL and decompositional approach already described. I also lay a notational foundation for describing probabilistic models and motivate the CRF family of models considered.

In Chapter 3, I specifically advertise decompositional semantic prediction to an SRL audience using formalisms, models, and features from that field. I establish a simple formulation of the problem and a benchmark evaluation which it revisited throughout the remainder of the thesis. The chapter also addresses the challenge of modeling structure between properties by using conditional random fields.

Chapter 4 presents a pytorch-based CRF modeling and inference library and then address weaknesses of our Chapter 3 evaluation, focusing on only a subset of the properties that enables me to reconsider the impact of label binarization and inter-property factors when exact inference is tractable. To motivate joint probabilistic decompositional models, I introduce the conditional SPR task which evaluates the model's ability to predict some properties conditioned on others.

Chapter 5 gives a novel presentation of ordinal models within the CRF framework. I investigate the impact of the ordinal inductive bias on the quality of the decomp model.

Chapter 6 revisits previous models using approximate inference, introducing features of the †x model for generalized loopy belief propagation on cluster graphs, comparing the impact of approximate inference at test time and during training, and exposing an issue with approximate loglikelihood-based training.

Chapter 7 reviews issues involved with multiple-annotator crowd data, including a review of ideas from the literature and preliminary concepts and experiments with both *apriori* and *aposteriori* label aggregation.

In Chapter 8, I conclude and mentions possible future directions.

# Chapter 2

# Background and Related Work

## 2.1 Introduction

In Chapter 1 I looked at the idea of semantic prediction as attempting to mimic human inferences about entities and events evoked by language. I also described the *decompositional approach* to semantic prediction that identifies a few properties of interest (e.g. did an argument *instigate* the event represented by a given predicate or is a need for *medical assistance* being expressed by a sentence). Inputs are then labeled by human annotators on a graded scale that reflects the degree to which the binary property is likely to hold given the evidence provided by the input text. The decompositional models in this thesis are CRF-based probabilistic models of these decomposed properties. In this background chapter, I review a number of alternative semantic formulations, justify our choice of CRF models, and establish notation.

## 2.2 Semantic Frameworks

Our methods and experiments facilitate work in decompositional semantic prediction—computational prediction under a particular framework for semantic representation. To motivate these efforts, this section summarizes various related and competing semantic representation frameworks, allowing us to highlight the singular strengths of the decompositional approach which are closely connected to the difficulties with data collection, modelling, and prediction that this thesis addresses.

For further references regarding semantic representation frameworks, the reader may consider a survey article by Abend and Rappoport (2017) who provide an overview of many formalisms (including the decompositional approach used in this thesis) for representing the meaning of text, particularly highlighting a variety of types of semantic information that can be annotated. Schubert (2015) offers a more formal look at semantic representations.

### 2.2.1 SRL

See Chapter 1, Section 1.1.1.

### 2.2.2 QA-SRL

QA-SRL (He, Lewis, and Zettlemoyer, 2015; He, 2018) is an attempt to avoiding the need for expert annotation of semantic prediction—one that potentially complements the decompositional work motivating this thesis. QA-SRL seeks to annotate semantic structure in addition to role-like labels. Rather than instructing annotators about argument identification and roles, QA-SRL presents a crowd annotator with a sentence $s$ and predicate $v$ and asks the worker to construct multiple template-based wh-questions[1] such that (1) the question contains $v$ and (2) the question can be answered using phrases (not necessarily contiguous in $s$. For our example (E2) from the introduction, questions and corresponding answers might be: “Who was led to the car?: the suspect”; “Who led the suspect to the car?: the officer”; and “Where was the suspect led?: to the car”.

The role labels obtained through this process are similar to the verb-specific roles of PropBank and, indeed, the authors give heuristic rules to automatically map their question-based roles to PropBank roles. Thus, they are able to help non-experts effectively label argument attachment.

The crowd-source protocol cannot guarantee recall of question-answer pairs, so FitzGerald et al. (2018) annotates a much larger set of data, trains a neural model, and uses high-recall predictions from the model as proposals for human validation in order to improve the recall of the crowd-sourced annotations.

Michael et al. (2018) propose QAMR—an extension of this approach beyond verbs and beyond the requirement of questions to follow a template. The result is a more complete set of questions representing the meaning of a sentence. However, since workers must highlight the answer within the sentence, the types of properties captured by the decompositional semantic approach are not generally represented in the QAMR formalism.

[1] Wh-questions are questions that start with one of the following words or phrases: “who”, “what”, “when”, “where”, “why”, “how”, “how much”.

### 2.2.3 Semantic Dependency Parsing

Oepen et al. (2014) defines Broad-Coverage Semantic Dependency Parsing (SDP) as "the problem of recovering sentence-internal predicate-argument relationships for all content words." The meta-formalism requires that all content words of a sentence be connected into a single graph using directed, labeled edges between lexical tokens. Tokens may additionally be labeled with word form, optional lemma, part of speech, a Boolean flag indicating whether the node represents a *top* predicate (possibly one of many), and an optional frame or sense tag.[2] While *syntactic* dependency parses also use bi-lexical, labeled edges to connect all lexical tokens of a sentence into a single tree structure, SDP only requires *content* words to be connected and has no tree or even acyclicity requirements on the graph structure. Rather than dictating a single graph-structured formalism for semantic representation, SDP data provides four target representations that all conform to the meta-formalism requirements. Stanovsky and Dagan (2018) suggest a linearization of the more general graph structure embodied by these parses and learn sequence-to-sequence models for translating to each of these representations from raw text or from one of the other three representations. Dozat and Manning (2018) modify a successful but simple architecture for neural syntactic dependency parsing which computes a score for each directed pair of tokens and then chooses a tree using a maximum spanning tree algorithm (a similar neural model scores the labels for each potential directed pair and chooses the label with the highest score). To make this syntactic model amenable to more general graph-based parsing they simply drop the spanning tree constraint and instead take *all* edges with positive score.

The most recent release from the LDC at the time of writing includes a dependency form

[2] See also Oepen et al. (2015) and Oepen et al. (2016).

of Combinatory Categorical Grammar parses. SRL constitutes a *partial* SDP formalism since it does not connect all content words.

### 2.2.4 Semantic Parsing with CCG

Combinatorial Categorical Grammar (CCG) (Ades and M. J. Steedman, 1982; M. Steedman, 2019) jointly models syntax and semantics. The syntactic expressiveness is known as "mildly context-free" and the semantic formalism is the typed lambda calculus introduced by Alonzo Church (Church, 1940). Categories in CCG include a syntactic component made up of non-terminal symbols combined with backward and forward slashes to refine possible contexts and a semantic component which is a logical form from the typed lambda calculus. The logical constants, which represent a combination of entities and predicates, are not formally defined but are typically taken as words from the input sentence. A CCG consists of a lexicon of possibly many categories for each token, a small set of rules for combining adjacent categories, and (optionally) a set of feature extractors and corresponding weights for scoring CCG parses.

While CCG is superficial in its representation of entities and predicates, it is explicit and exhaustive in its model of the composition of lexical units into a complete sentence. In contrast, decompositional semantics represent a more shallow model of composition and a more thorough and explicit model of entity and event meaning.

### 2.2.5 Sentence and Document Embeddings

Our prediction can make use of more generic semantic *vector* models. These models leverage a variety of bulk unsupervised or already available semantically annotated data to find meaningful generic representations for word tokens and for composition of those tokens into sentence representations. For example, the vectors produced by Devlin et al. (2018) represent a widely used and extended method of this type.

## 2.3 Modeling and Inference with CRFs

The core of the decompositional semantic models that I leverage in this thesis can be formulated as Conditional Random Fields (CRFs). CRFs, introduced by Lafferty, McCallum, and Pereira (2001), are a generalization of logistic regression models as a way to model multiple interdependent outputs conditioned on observed features of the inputs. CRFs are expressive, intuitive, and admit efficient inference and learning. Because I will use this formulation repeatedly, I summarize the idea here and establish notation.

### 2.3.1 MRFs

CRFs are closely related to Markov Random Fields (MRFs).[3] In both CRFs and MRFs, there is a collection of variables being modeled. For example, the main variables being modeled for the decompositional tasks in this thesis are the responses to questions about the role of an argument in a given predicate. A graph in which the nodes represents variables

[3] Note that I posted an earlier draft of some of the following as an answer on “Cross-Validated” (A. Teichert, 2019).

being modeled can be used to indicate aspects of the model—in particular, properties of conditional independence between subsets of variables.

Formally, an MRF with respect to an undirected graph $G$ is simply

1. a set of random elements (a.k.a random "variables") corresponding to the nodes in $G$

2. with a joint distribution that is *Markov* with respect to $G$; that is, the joint probability distribution associated with this MRF is subject to the following "Markov constraint" given by G:

   For any two variables, $V_i$ and $V_j$, the value of $V_i$ is conditionally independent of $V_j$ given its neighbors $\mathcal{B}_i$. In this case, it is said that the joint probability distribution $P(\{V_i\})$ *factorizes* according to $G$.

## 2.3.2 CRFs

In contrast, a Conditional [Markov] Random Field (CRF) with respect to a graph $G$ only models the joint distribution of a fixed *subset* of the variables, conditioned on the remaining variables. More formally,

**Definition 2.3.1.** Given an undirected graph $\mathcal{G} = (\mathcal{I}_X \cup \mathcal{I}_Y, \mathcal{E})$. A *Conditional Random Field* with respect to $\mathcal{G}$ is a set of random variables $\{X_i\} \cup \{Y_i\} = \{V_i\}_{i \in \mathcal{I}_X \cup \mathcal{I}_Y}$ with conditional distribution $P(\{Y_i\}|\{X_i\})$ that is Markov with respect to $\mathcal{G}$.

Any such distribution can be defined in terms of the product of clique-specific factor functions $\{\Psi_a\}_{a \in A}$ where $A$ is the set of maximal cliques in $G$ and $\boldsymbol{y}_a = \{\boldsymbol{y}_i\}_{i \in a}$:

$$P(\{Y_i\} = \boldsymbol{y} | \{X_i\} = \boldsymbol{x}) :\propto \prod_{a \in A} \exp\left(\Psi_a(\boldsymbol{y}_a, \boldsymbol{x})\right) \tag{2.1}$$

Given an assignment (or "configuration") $\boldsymbol{y}_a$ of each of the variables $V_i \in a$, each potential function $\Psi_a$ assigns a real-valued score as a function of the variable configuration $\boldsymbol{y}_a$ and the input $\boldsymbol{x}$ (higher scores corresponding to higher probability as indicated in the equation above). (If we disallow the extra arbitrary dependence on $\boldsymbol{x}$ we simply have an MRF.)

Since a CRF does not need to obey Markov constraints on the observed variables $\{X_i\}$, these observed variables are typically not even shown in graphical representations of a CRF. Also, although any distribution can be captured using only potential functions on maximal cliques, it is often convenient to include factors on non-maximal cliques as well. This can be used to clearly indicate where parameters of the model are reused across multiple cliques.

Potential functions that do not look at $\boldsymbol{x}$ and that only deal with discrete, finite variables can simply be represented as real-valued tensors—each cell holding the function value corresponding to the respective variable configuration (e.g. real-valued vectors for unary factors and matrices for pairwise factors). Probabilistic inference can then be carried out using a sequence of tensor operations to appropriately marginalize out observed variables.[4] Indeed, it can be convenient to view a CRF model as simply specifying a set of cliques $A$, and corresponding functions from input $\boldsymbol{x}$ to MRF potential function parameters, e.g. the tensors corresponding to each $\Psi_a$. Often, parametric functions are used so that the model

[4] Though exact inference may require an exponential runtime.

can be tuned on labeled data.

### 2.3.3 Factor Graphs

*Factor graphs* (Frey et al., 1997) are convenient for expressing the set of actual cliques used in a MRF or CRF model. A factor graph is a bipartite graph where each random variable $V_i$ is represented by a *variable* node and each potential function $\Psi_a$ is represented by a *factor* with undirected edges between that node and each participating variable $V_i \in a$. Note that there may be multiple factors touching the same set of variables.

The visual representation of a factor graph is a bipartite graph with a circular node for each random variable, a square node representing each factor function, and an edge between factors and the variables that are part of the respective factor's domain.

### 2.3.4 Factor Templates

Factor parameters can often be shared across many factors. For example, when reasoning about the sequence of part-of-speech tags for a given input sentence of length ten, we could try to capture the idea that there is consistency in transition probabilities from the tag on one word to the tag on the next word of the sentence. Therefore, we could use the same potential function on each of nine "adjacent-tag" factors. Thus, the same parameters would be used (and tuned) to model the distribution over possible tag sequences regardless of the length of the input. Compare this to "word-emission" factors that specify the preference for the various tags as a function of observed features of the the word at each, respective position. For example, observing that the word at a given position ends in "er" could correspond to a separate, learned weight for each of the possible tag labels. The emission factor at positions with that feature would be impacted by those shared weights. Other such *observation features* could include the number of characters in the word, or whether the word appears in a particular list of words. Given training data, *feature templates* such as "last-three-letters" can give rise to many observation features which in term correspond to shared parameters.

A *factor template* is simply a reusable specification of how to combine model parameters and observation features to determine the potential function for a particular factor.

To summarize, it is common to specify a parametric CRF distribution family by providing a function from input to a generated list of (already conditioned) MRF factors. Each factor identifies a set of variables (possibly overlapping between factors) and a potential function. Factors are often associated with a factor template and parameters of the potential function are computed as a function of model parameters and some subset of the available input. Observation features relevant to a particular template can likewise be specified in terms of features templates which means that the actual features attended to and the number of features/parameters may be determined at training time based on characteristics observed in the training data.

### 2.3.5 Region Graphs and Belief Propagation

Often, the goal of a model is to compute one or both of the following:

1. The normalizing constant from Equation 2.1.

2. The marginal probability of some partial configuration of variables.

Given an MRF factor-graph with potential functions, the *unnormalized* probability of any *full* configuration is easily computed by applying each factor function to the participating subset of the joint configuration and multiplying the resulting function values. However, to compute the actual probability of a joint configuration or to compute other marginal inference queries requires summing this product of factors over all possible joint configurations (of which there are an exponential number). Fortunately, if there are no loops in the factor-graph representation, then the sum-product algorithm (which is a slight generalization of the forward-backward algorithm and also known as *belief propagation*) can exactly compute this normalizing constant or exact marginal probabilities of each variable in runtime that is linear in the size of the graph. Similarly, the max-product algorithm can be used to find a joint assignment that maximizes the joint conditional probability.

Unfortunately, when the factor graph is not a tree, cycles arise in the update definitions and guarantees of convergence and correctness are lost. There ia a large amount of work on improving the convergence and approximation performance of belief propagation and variants.

I leverage generalized belief propagation via the parent-to-child algorithm on region graphs (Yedidia, Freeman, and Y. Weiss, 2005).[5] The approach is sufficiently general as to subsume vanilla belief propagation and a generalization for cluster graphs which admits exact inference on cyclic graphs via the junction tree method. Despite the generality, it maintains the basic flavor and simplicity of the original BP message passing algorithm and allows experimentation with message prioritization and pruning for further potential improvements.

[5] I also appreciate the description of the same work provided by Welling (2004).

Given a factor graph, a region-based approximation for inference on the factor graph is specified as a set of regions $\mathscr{R}$: each region, $r \in \mathscr{R}$ being assigned a "counting number" $c_r \in \mathbb{R}$ and a set of nodes from the factor graph. If a factor node is included in region $r$, then so must all of the variables it touches.

To minimize over- or under-counting, a "valid" region-based approximation requires that the total counting for any node must be one:

$$\sum_{r \in \mathscr{R} s.t. i \in r} c_r = 1 \qquad \forall i \tag{2.2}$$

The free energy (i.e. the log normalizing constant) of the factor graph is then approximated as the weighted sum of region free energies.

To use the "region graph" method, I additionally connect the regions with an acyclic set of directed edges $\{st \in E\}$ with the constraint that $\forall_{st \in E} s \supset t$. Such a region admits use of the "parent-to-child" message-passing algorithm for inference (Yedidia, Freeman, and Y. Weiss, 2005; Welling, 2004). Other work has also looked at the impact of various choices of regions (Welling, 2004). Given current variational parameters (or "message") which can be initialized uniformly or randomly, approximate region marginals (or "beliefs") are given as follows:

$$b_r(x_r) = \frac{1}{Z_r} \prod_{a \in F_r} f_a(x_a) \prod_{st \in S_r} m_{st}(x_t) \tag{2.3}$$

Vanilla BP imposes *consistency* constraints that the belief (i.e. approximate marginal) for individual variables should agree with the beliefs computed for the factors that the variable participates in. More generally, we can require that the beliefs at target regions agree with beliefs of shared variables for source regions. These consistency constraints give rise to the general message update rule of Equation 2.8:

$$b_t(x_t) = \sum_{x_{s\setminus t}} b_s(x_s) \qquad \forall st \in E, \forall x_t \tag{2.4}$$

$$\mathbf{1} = \frac{\sum_{x_{s\setminus t}} b_s(x_s)}{b_t(x_t)} \qquad \forall st \in E, \forall x_t \tag{2.5}$$

$$\mathbf{1} = \frac{\sum_{x_{s\setminus t}} \frac{1}{Z_s} \prod_{a\in F_s} f_a(x_a) \prod_{ij\in S_s} m_{ij}(x_j)}{\frac{1}{Z_t} \prod_{a\in F_t} f_a(x_a) \prod_{ij\in S_t} m_{ij}(x_j)} \qquad \forall st \in E, \forall x_t \tag{2.6}$$

$$\mathbf{1} = \frac{Z_t}{Z_s} \frac{\sum_{x_{s\setminus t}} \prod_{a\in F_s} f_a(x_a) \prod_{ij\in S_s} m_{ij}(x_j)}{\prod_{a\in F_t} f_a(x_a) m_{st}(x_t) \prod_{ij\in S_t - st} m_{ij}(x_j)} \qquad \forall st \in E, \forall x_t \tag{2.7}$$

$$m_{st}(x_t) \propto \frac{\sum_{x_{s\setminus t}} \prod_{a\in F_s f_a(x_a)} \prod_{ij\in S_s} m_{ij}(x_j)}{\prod_{a\in F_t f_a(x_a)} \sum_{x_{s\setminus t}} \prod_{ij\in S_t - st} m_{ij}(x_j)} \qquad \forall st \in E, \forall x_t \tag{2.8}$$

where $F_r$ is the set of factor nodes in $r$, $S_r$ is the set of edges $\{ij \in E\}$ from a region $i$ that is not reachable from $r$ to a region $j$ that is reachable from $r$.

Note that, although the region graph forms a DAG, the message updates may still have cyclic dependencies in general. In the general case, therefore, messages are updated according to some message-passing schedule until stopping criteria are met.

The resulting beliefs can be treated as approximate marginals; however, if message passing has not converged or if there are disconnected subsets of the region graph dealing with the same factor-graph node, then these beliefs might not agree with respect to these approximate marginals.

The normalizing constant can also be approximated using the resulting beliefs. The basic idea is that if a single configuration's true probability were known, that configuration could be easily scored as a product of scalars which would immediately reveal the normalizing constant by dividing the score by the probability. If we had time to enumerate configurations, we could compute $Z$ in this way for each configuration. A weighted geometric mean of these identical values would likewise result in the same value $Z$. If the weights sum to $1$ (as the true probabilities do), the expression simplifies and yields a way to compute $Z$ that decomposes into two terms—the first factorizes exactly over the factors of the distribution while the second is the entropy of the distribution which approximately factorizes additively over "regions" of the distribution:

$$b(x) = \frac{\prod_{a\in F} f_a(x_a)}{Z} \tag{2.9}$$

$$Z = \frac{\prod_{a\in F} f_a(x_a)}{b(x)} \tag{2.10}$$

$$= \exp \sum_x b(x) \log \frac{\prod_{a\in F} f_a(x_a)}{b(x)} \tag{2.11}$$

$$\log Z = \sum_x b(x) \log \prod_{a\in F} f_a(x_a) - \sum_x b(x) \log b(x) \tag{2.12}$$

$$\log Z = \sum_x b(x) \sum_{a\in F} \log f_a(x_a) - \sum_x b(x) \log b(x) \tag{2.13}$$

$$\log Z = \sum_x \sum_{a\in F} b(x) \log f_a(x_a) - \sum_x b(x) \log b(x) \tag{2.14}$$

$$\log Z = \sum_{a\in F} \sum_x b(x) \log f_a(x_a) - \sum_x b(x) \log b(x) \tag{2.15}$$

$$\log Z = \sum_{a\in F} \sum_x b_a(x_a) b_{\neg a}(x_{\neg a}) \log f_a(x_a) - \sum_x b(x) \log b(x) \tag{2.16}$$

$$\log Z = \sum_{a\in F} \sum_{x_a} \sum_{x_{\neg a}} b_a(x_a) b_{\neg a}(x_{\neg a}) \log f_a(x_a) - \sum_x b(x) \log b(x) \tag{2.17}$$

$$\log Z = \sum_{a\in F} \left( \sum_{x_{\neg a}} b_{\neg a}(x_{\neg a}) \right) \sum_{x_a} b_a(x_a) \log f_a(x_a) - \sum_x b(x) \log b(x) \tag{2.18}$$

$$\log Z = \sum_{a\in F} \mathbf{1} \sum_{x_a} b_a(x_a) \log f_a(x_a) - \sum_x b(x) \log b(x) \tag{2.19}$$

$$\log Z = \sum_{a\in F} \sum_{x_a} b_a(x_a) \log f_a(x_a) - \sum_x b(x) \log b(x) \tag{2.20}$$

The “free energy” of the region graph distribution is the negative log partition function. The decomposition above allows us to define a region-based approximation as follows:

$$\mathcal{F}_{\mathcal{R}} = \sum_{r\in\mathcal{R}} c_r \mathcal{F}_r \tag{2.21}$$

where the free energy of an individual region is given as follows:

$$\mathcal{F}_r = \sum_{x_r} b_r(x_r) \log b_r(x_r) - \sum_{x_r} b_r(x_r) \sum_{a \in F_r} \log f_a(x_a) \tag{2.22}$$

Because multiplication distributes over addition, messages into the receiving region can either be excluded from the product before the summation or can be divided out afterward. Since zeros will eventually be propagated, the only time that zeros occur in the denominator is if they would eventually be multiplied in, so any would-be zeros in the denominator can be replaced by 1s.

The divide-out version can be helpful since it allows multiple outgoing messages from the same region to be computed simultaneously—sharing the work of multiplying all messages into the region; then the product can be separately summed to achieve outgoing message that each would divide out a separate set of messages. In some cases, jointly multiplying and summing is done sparsely (as in structured factors).

### 2.3.6 Reparameterization

Message passing can be seen as a process of discovering a reparameterization of the distribution that is more convenient in the following two ways:

1. It immediately reveals approximate marginals (or max-marginals) for each factor (or even clusters of factors).

2. It is approximately normalized so that the product of the factors is roughly equal to

(rather than merely proportional to) the probability. (Or sometimes, the normalizing constant is itself useful.)

If we knew the marginals at each factor, we could construct a reparameterization that accomplishes (1) by multiplying factor potentials by appropriately chosen "message" factors, arriving at the true factor marginals (I will call these the "belief" factors) and then, so as to not change the product, additionally including the reciprocals of these messages as additional "inverse message" factors.

Similarly, this reparameterization of the same distribution via marginals and messages would allow us to compute an approximation to the normalizing constant.

$Z = \sum_y \prod_\alpha f_\alpha(y_\alpha)$.

$$\begin{aligned}
0 = D_{\text{KL}}(b||p) &= \sum_y b(y) \ln \frac{b(y)}{p(y)} \\
&= \sum_y b(y) \ln \frac{b(y)}{\frac{1}{Z}\prod_\alpha f_\alpha(y)} \\
&= \sum_y b(y) \ln b(y) - \sum_y b(y) \ln \frac{1}{Z} - \sum_y b(y) \prod_\alpha f_\alpha(y_\alpha) \\
&= \sum_y b(y) \ln b(y) + \ln Z - \sum_y b(y) \sum_\alpha \ln f_\alpha(y_\alpha) \\
&= \ln Z + \sum_y b(y) \ln b(y) - \sum_y b(y) \sum_\alpha \ln f_\alpha(y_\alpha)
\end{aligned}$$

,

Threfore:

$$\begin{aligned} -\ln Z &= \min_b \sum_y b(y)\ln b(y) - \sum_y b(y)\sum_\alpha \ln f_\alpha(y_\alpha) \\ &= \min_b \sum_y b(y)\ln b(y) - \sum_y b(y)\sum_\alpha \ln f_\alpha(y_\alpha) \\ &= -S(x) - U(x) \end{aligned}$$

Since the original unnormalized distribution factorizes across regions and if we assume that the beliefs give the correct marginals at each region, then we can compute the last term exactly (if beliefs are truly marginals):

$$\begin{aligned} U(x) &= \sum_y b(y)\sum_\alpha \ln f_\alpha(y_\alpha) \\ &= \sum_\alpha \sum_{y_\alpha} b_\alpha(y_\alpha)\ln f_\alpha(y_\alpha) \end{aligned}$$

However, we cannot, in general, efficiently compute $S(x)$ exactly since the total (normalized) belief is not just the product of normalized region beliefs. Nevertheless, since we can compute the normalizing constant of each region exactly, I approximate the full function by combining region-based constants in such a way that ensures that each factor and variable get counted only once.

## 2.4 Conclusion

In this chapter I have situated our target decompositional semantic framework among many alternative frameworks and introduced our primary method and notation for inference and learning.

# Chapter 3

# Multi-label Binary Semantic Decomposition[1]

## 3.1 Introduction

Semantic (thematic) roles are traditionally labeled with nominal categories such as “Agent”, “Patient”, and “Theme” (e.g. Baker, Fillmore, and Lowe (1998)). Such categorization is rooted in semantic theory and has been instrumental in early generations of natural language semantic processing by computer, being amenable to systematic annotation of large corpora analogous to the Penn Treebank effort (Marcus, Marcinkiewicz, and Santorini, 1993). Unfortunately, such categorization efforts also have significant shortcomings which become increasingly pronounced as our field recognizes and addresses the challenges of natural language processing in a wider diversity of domains and languages.

[1] The contents of this chapter are presented here with some adaptation from work published in A. Teichert et al. (2017). See: Acknowledgments.

First, nominal categorization fails to indicate degrees of similarity between instances with differing labels and differences between instances with the same label. Decomposing categorical labels into multi-dimensional binary labels reveals more nuanced relationships between labels, and these relationships can be exploited by structured models. Our multi-label binary model capitalizes on this structure within a familiar binary framework for inference while conveniently side-stepping the fact that some examples may be non-comparable along certain dimensions.

Dowty (1991) argued against the categorical notion of semantic (thematic) roles, suggesting instead a multi-faceted relationship between an argument and a predicate which he termed *proto-roles*. He replaced traditional categories such as AGENT or PATIENT with prototypical assumptions of underlying semantic properties; e.g. a PROTO-AGENT is likely to be *aware* and *volitional*. Kako (2006) found additional evidence to support Dowty's theory, and Reisinger et al. (2015) subsequently constructed a dataset supporting the task of semantic proto-role labeling (SPRL): predicting human responses to questions on individual properties. For an illustration, I invite the reader to consider in the following examples, semantic properties of what was *led*: Was the argument aware of being led? Was it sentient? Was it willing? Did it instigate the leading?

a) The officer *led* **the convict** to the car.

b) California *led* **the nation** in sales.

c) The guide *led* **John** past the danger.

The SPRL task pursued in this chapter is a departure from PropBank (Palmer, Gildea, and Kingsbury, 2005) semantic role labeling (SRL) which would annotate all of the above examples with the same verb sense (LEAD.01) and argument role (ARG1). The SPRL

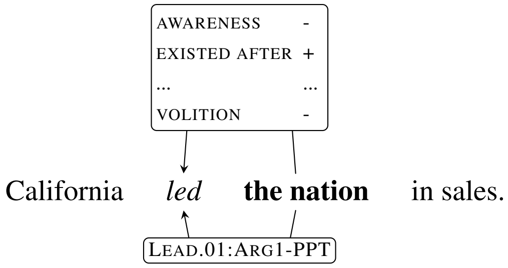


**Figure 3.1.** SPRL (top) VS SRL (bottom).

questions, however, distinguish between these examples without assigning a categorical label. In addition to annotational benefits of such a decompositional approach, I expect this contrast to provide an opportunity for synergistic joint modeling of SPRL and SRL.

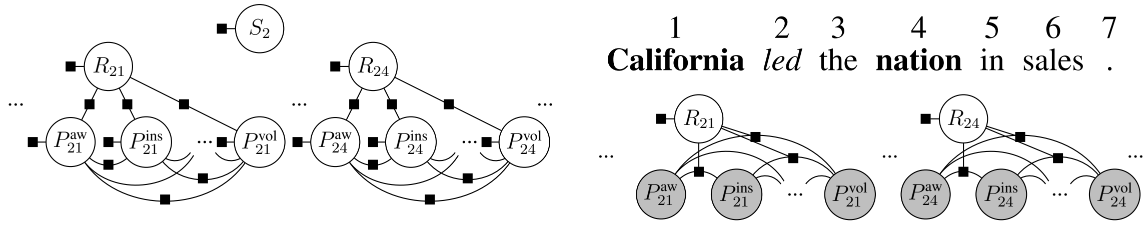
In what follows I:

- specify a multi-label classification evaluation for SPRL appropriate for joint labeling of entire input sentences;
- establish a strong SPRL result backed by an SRL model with reasonable performance on a standard dataset;
- evaluate a variety of models with SPRL *and* SRL;
- report SRL results on PropBank semantic-function tags for Ontonotes 5, contrasting the tagset to PropBank numeric (ArgN) labels via their impact on our models.

## 3.2 Tasks

Our contributions provide a bridge to encourage people with experience in SRL or semantic role inventories to consider how they could bring their expertise to bear on a decompositional approach.

Figure 3.1 demonstrates the varieties of semantic labeling that I explore in this chapter.



**(a)** SRL+SPRL⋆ with sense variables

**(b)** SRL | SPRL⋆: uses pairwise features of observed properties

**Figure 3.2.** Factor graphs depicting two models instantiated on the sentence from Fig. 3.4. $S_2$ is a sense variable to identify a PropBank frame for the predicate *led*. $R_{21}$ is the SRL variable for the role of **California** with respect to the predicate. $P_{21}^{\text{aw}}$,$P_{21}^{\text{ins}}$ and $P_{21}^{\text{vol}}$ are binary variables representing whether or not **California** has respectively awareness, instigation, and volition in the *led* event ($P_{ij}^{q} \triangleq P_{ijq}$). $R_{24}$ is the role variable for **nation** in the *led* event.

SRL is traditionally a *multi-class* classification problem where predicate-argument pairs are assigned a label describing the role of the argument in the event. I investigate two label sets for SRL. *ArgN* labels (e.g. Arg0, Arg1) associate the argument to numbered slots for the particular predicate (although the numbers tend to hold similar meaning across predicates, but this is not guaranteed). The semantic function tags (*SFT*) of Bonial, Stowe, and Palmer (e.g. PPT, GOL) associate the argument with one of a small set of coarse-grained roles that have meaning across all predicates.

I also investigate proto-role models. I cast SPRL as a *multi-label* classification task where each (pre-identified) predicate-argument pair is assigned a *set* of properties (e.g. {awareness, volition}). For each property, Reisinger et al. asked annotators to judge on a five-way Likert scale (i.e. from 'very unlikely' to 'very likely'), how likely it is that the property holds in the given context. In the case where the response fell below the ordinal level of 'likely', annotators were asked a follow-up binary question of whether or not the property is applicable for the given predicate-argument pair in context. Since our primary goal is to establish results for SPRL that are easily interpreted and compared against, I choose to side-step the issue of applicability as well as the complicating aspects of ordinal labels, and instead formulate the prediction problem as multi-label classification. I let the gold label for each predicate-argument pair be the set of properties annotated as with a response of 4 or 5 (i.e. ‘likely’ or ‘very likely’).

## 3.3 Models and Features for SPRL and SRL

To establish strong results for SPRL and to advertise the task to relevant communities, I take inspiration from the related SRL models of Gormley et al. (2014). Intuitively, we can expect that features that effectively predict a categorical representation of semantic role should likewise be effective in modeling a decompositional approach to semantic role. I explore several models for three tasks: SRL alone, SPRL alone, and joint prediction of SRL and SPRL. I also optionally include predicate-sense prediction.

### 3.3.1 Formulation

Each model is formalized as a conditional random field or CRF (Lafferty, McCallum, and Pereira, 2001). For each given pred-arg pair $(i, j)$ and property $q$, I instantiate three types of variables: $P_{ijq}$ is a binary variable with labels $\{+, -\}$ representing that $q$ does or does not hold respectively. $R_{ij}$ is a multi-class variable ranging over SRL role labels. When only the predicate index $i$ is given, I instantiate $R_{ij}$ for all $j$ and allow the role label NIL to indicate that there is no semantic pred-arg relationship. $S_i$ is a multi-class variable ranging over the possible predicate senses. For each model, I select from these variables a task-specific subset:

$$\boldsymbol{Y} = \{Y_k\} \subseteq \{P_{ijq}\} \cup \{R_{ij}\} \cup \{S_i\}.$$

Given the input sentence $\boldsymbol{x}$, the probability of a joint assignment $\boldsymbol{y} = \{y_k\}$ to the variables $\boldsymbol{Y}$ is given by a globally normalized distribution:

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) \propto \prod_{a \in A} \exp\left(\boldsymbol{w}^T \boldsymbol{f}_a(\boldsymbol{y}_a, \boldsymbol{x})\right),$$

where each $a \in A$ is an index set of variables that some feature looks at jointly, $\boldsymbol{y}_a$ is the corresponding subset of $\boldsymbol{y}$, and $\boldsymbol{w}$ is a vector of parameters. In a factor-graph representation (Frey et al., 1997), $A$ corresponds to the set of factors and defines the independence assumptions.

### 3.3.2 Models

I define five models that vary in two key aspects: the types of variables I include and the structure of the graphical model, given by $A$.

- SRL includes role variables $\{R_{ij}\}$ with an independent multi-class logistic regression for each—a graphical model with only unary factors.
- SPRL includes property variables $\{P_{ijq}\}$ with an independent binary classifier for each conjunction of predicate-argument pair $(i, j)$ and property $q$.
- SPRL$^\star$ has the same variables as SPRL but allows for interactions between pairs of properties. For each pair of properties $q$ and $r$, there is a factor between $P_{ijq}$ and $P_{ijr}$.
- SRL+SPRL combines models SRL and SPRL by adding a factor between each SPRL property variable $P_{ijq}$ and its corresponding SRL role variable $R_{ij}$.

- SRL+SPRL⋆ is our full joint model and includes all factors from models SPRL⋆ and SRL+SPRL. See Figure 3.2.

The conditional models SRL|SPRL (SRL *given* SPRL) and SPRL|SRL are identical to SRL+SPRL, except that the gold value of each property variable $P_{ijq}$ or role variable $R_{ij}$ is observed respectively—likewise for SPRL⋆|SRL versus SRL+SPRL⋆. SRL|SPRL⋆ is identical to SRL|SPRL with the addition of indicator features for each $R_{ij}$ that look at observed *pairs* of SPRL properties.

When evaluating on sense prediction, I also include the sense variables $\{S_i\}$, although they do not share factors with the other variables of the models. I use belief propagation (Pearl, 1988; Kschischang, Frey, and Loeliger, 2001) for inference. For the models with cycles (SPRL⋆, SRL+SPRL⋆), I run loopy belief propagation (Pearl, 1988; Murphy, Yair Weiss, and Jordan, 1999) with a maximum of five iterations. Our implementation uses the Pacaya library(Gormley, 2015a)[2].

### 3.3.3 Features

As is typical in CRFs, I define each of our features on a factor $a$ as a conjunction of an indicator $\mathbb{1}\left[\right]$ for a fixed variable assignment $\tilde{\boldsymbol{y}}_a$ with some *observation-feature* function $g_{ak}$ of the input sentence:

$$f_{a,k,\tilde{\boldsymbol{y}}_a}(\boldsymbol{y}_a, \boldsymbol{x}) = \mathbb{1}\left[(\boldsymbol{y}_a = \tilde{\boldsymbol{y}}_a)\right] g_{ak}(\boldsymbol{x}).$$

I include over one hundred observation-features motivated by prior work in dependency-based SRL (Björkelund, Hafdell, and Nugues, 2009; Zhao et al., 2009; Lluís, Carreras, and

[2] https://github.com/mgormley/pacaya

| | **annotated** | **pred-arg** | **# label types** | |
|---|---|---|---|---|
| | **sentences** | **instances** | ArgN | SFT |
| CoNLL09 | 43,012 | 430,850 | 53 | - |
| OntoFull | 35,497 | 266,298 | 31 | 26 |
| OntoMed | 24,755 | 185,878 | 31 | 26 |
| OntoSmall | 4,912 | 36,618 | 27 | 24 |
| PropSmall | 4,912 | 9,738 | 20 | 16 |

**Table 3.1.** Dataset sizes

Lluís Màrquez, 2013). The features use the sentence's words, lemmas, Brown clusters (Brown et al., 1992)[3], part-of-speech tags, and syntactic dependency parse. When present, inter-property and SPRL-SRL factors only include a bias parameter for each configuration. I employ the feature-hashing trick (Ganchev and Dredze, 2008; Weinberger et al., 2009) to restrict the number of model parameters.[4]

Prior work has explored joint syntactic and semantic dependency parsers to understand the interaction between the two linguistic strata (Johansson, 2009; Gesmundo et al., 2009; Naradowsky, Riedel, and Smith, 2012; Lluís, Carreras, and Lluís Màrquez, 2013; Gormley et al., 2014). Here, by contrast, I am interested in the relation between different semantic annotation schemes. Nonetheless, our joint model is similar in both form and features.[5]

**Roleset:** *Lead.01*
**Name:** *directed motion, be ahead of*
Arg0-**PAG**: leader
Arg1-**PPT**: in the lead of
Arg2-**EXT**: extent
Arg4-**DIR**: start point
Arg5-**GOL**: end point
**Examples:**
*cause to go*: John led the unhappy ...
*go before*: California led the nation ...
...

**Figure 3.3.** Example of information available in PropBank framesets (v3.1) for Lead.01.

# 3.4 Experiments

## 3.4.1 Datasets

Our experiments use several datasets. PropBank adds semantic role labels to the syntactic annotations available on the Wall Street Journal (WSJ) portion of the Penn Treebank (Marcus, Marcinkiewicz, and Santorini, 1993). Each predicate instance in the corpus is labeled with a *verb sense* (a.k.a. *roleset*) which has a corresponding *frame*. See Figure 3.3 for the frame corresponding to LEAD.01 from our example. Each frame describes the *slots* that can be filled by the predicate's arguments. Arguments of each predicate instance are identified as such and labeled so as to identify which slot it fills. For example, California fills the ARG1 slot in Figure 3.1. ***PropSmall*** contains the subset of PropBank predicate-argument pairs as filtered and further annotated by Reisinger et al. (2015). This is our only dataset containing SPRL annotations.

[3] I use `https://github.com/percyliang/brown-cluster` to create 1000 clusters of wikitxt from the polyglot project (Al-Rfou, Perozzi, and Skiena, 2013). Our features look at the full id and length-five prefixes.

[4] In particular, I hash each observation feature $g_{ak}(\boldsymbol{x})$ to a number between 0 and the prime number 1,000,003 and conjoin the resulting feature id with the factor type and label category so that features $f_{a,k,\tilde{\boldsymbol{y}}_a}(\boldsymbol{y}_a, \boldsymbol{x})$ for different factor types and configurations do not overlap.

[5] The model of Naradowsky, Riedel, and Smith looks especially similar to ours for SPRL (i.e. they include a collection of binary variables for each pred-arg pair); however, theirs is a *multi-class* model using hard factors

In PropBank, the role labels (e.g. Arg1, Arg2) are not necessarily consistent in meaning across rolesets and must be disambiguated by the frame. However, Ontonotes 5 (Weischedel et al., 2013; Bonial, Stowe, and Palmer, 2013) — a more recent extension of PropBank[6] — additionally annotates each slot with one of a small number of labels called *propbank semantic function tags* (SFT) whose meanings are not roleset specific. These are shown after the hyphen in the example of Figure 3.3. Having roleset-independent tags justifies sharing statistical strength of observations across all training examples. The Ontonotes 5 dataset includes most (but not all) of the Penn Treebank WSJ sentences as well as data from other genres. Our experiments on Ontonotes 5 are restricted to the WSJ subset. ***OntoFull*** is composed of all overt predicate-argument pairs in the WSJ portion of Ontonotes 5. It includes SFT annotations in addition to ArgN SRL labels. ***OntoMed*** and ***OntoSmall*** include the pairs from random subsets of the sentences in OntoFull. Figure 3.1 compares the sizes of our datasets.

The PropBank, Ontonotes, and SPRL datasets were originally annotated relative to constituency parses. I automatically map gold constituency parses to universal Stanford dependencies (Marneffe et al., 2014) and gold part-of-speech tags to the universal part-of-speech tagset (Petrov, Das, and McDonald, 2012). [7]

***CoNLL09*** is the English SRL data from the CoNLL-2009 shared task (Hajič et al., 2009; Surdeanu et al., 2008) and includes verbal and nominal predicates from PropBank (Palmer,

to enforce mutual exclusion of the labels and is more akin to our SRL model. Such constraints are inappropriate for multi-label SPRL.

[6] I used the release-candidate version of the frames: https://github.com/propbank/propbank-frames/tree/release-candidate

[7] I use PyStanfordDependencies: https://github.com/dmcc. As the gold head for PropBank and OntoNotes predicates and arguments, I select the left-most token whose parent in the converted gold dep-parse is not in the set of dominated tokens.

Gildea, and Kingsbury, 2005) and NomBank (Meyers et al., 2004) respectively. The English data from the CoNLL-2009 shared task (Hajič et al., 2009; Surdeanu et al., 2008) included head-based semantic role labeling and sense prediction. I use the CoNLL-2009 data to validate the performance of our SRL model.

### 3.4.2 Training

I train our models using stochastic gradient descent (SGD) with the AdaGrad adaptive learning rate and a composite mirror descent objective with $\ell_2$ regularization following Duchi, Hazan, and Singer (2011). I used the train data to define the SGD objective and to (optionally) adjust the AdaGrad $\eta$ parameter during learning (Bottou, 2012). I used our evaluation objective (e.g. Labeled SPRL F1) on the dev data for early stopping.[8] Wherever I report aggregated F1 over all properties, it is micro-averaged F1 (i.e. statistics are aggregated across categories and then precision, recall and f1 are computed once on the aggregate stats). I used random search for hyper-parameter optimization (Bergstra and Bengio, 2012), sampling thirty random configurations.[9] For each model scenario, I trained under all hyper-parameter configurations, selected the model with the best dev performance, and evaluated on held out data.

Optimization uses a seed for the random number generator which is set arbitrarily to 3 for all but the conll experiments. With the exception of CoNLL09, I split the datasets on

[8] Our joint models were each trained in view of optimizing only one objective at a time. That is, the models in Table 3.6 were trained using labeled SRL accuracy as the evaluation objective while the models in Table 3.8 used SPRL Property F1.

[9] For each random configuration, hyper-parameters were independently selected from the following ranges: adaGradEta [5e-4, 1.0], L2Lambda [1e-10, 10], featCountCutoff {1,2,3,4}, sgdAutoSelectLr {True, False}. Continuous parameters were sampled on a log scale and then rounded to 2 significant digits.

| | **SRL +sense +arg-id Labeled F1** | |
|---|---|---|
| 0 | Naradowsky, Riedel, and Smith (2012) | 78.55 |
| 1 | Gormley et al. (2014) | 86.54 |
| 2 | our SRL model | **87.40** |
| | **SRL +sense Accuracy** | |
| 3 | our SRL model | 90.78 |
| | **SRL Accuracy** | |
| 4 | our SRL model | 91.48 |

**Table 3.2.** English CoNNL 2009 SRL given gold syntax. Lines 0-1 report published results evaluating labeled role and sense F1 with predicate heads pre-identified; line 2 is our model for the same setting, the line 3 model has predicate-argument pairs pre-identified (so F1=Accuracy), and line 4 drops sense disambiguation from the evaluation.

| | **SRL +sense +arg-id F1** | |
|---|---|---|
| 0 | Naradowsky, Riedel, and Smith (2012) | - |
| 1 | Gormley et al. (2014) | - |
| 2 | our SRL model | **84.30** |
| | **SRL +sense Accuracy** | |
| 3 | our SRL model | 88.30 |
| | **SRL Accuracy** | |
| 4 | our SRL model | 90.34 |

**Table 3.3.** Dev results (where available) corresponding to table 3.2.

WSJ section boundaries as follows: train (0-18), dev (19-21), test (22-24). To compensate for the smaller size of the PropSmall dataset which was filtered and sampled from PropBank by Reisinger et al., our split reserves a larger proportion of the data for development and test than does CoNLL09. At various points during this dissertation, I have found it useful to be able to compare against development set results, so I include those results in addition to the held-out test-set results.

| | | OntoFull | | PropSmall | |
|---|---|---|---|---|---|
| | **train** | ArgN | SFT | ArgN | SFT |
| 0 | OntoFull | 88.3 | 87.5 | 86.1 | 82.9 |
| 1 | OntoMed | 87.2 | 86.5 | 85.8 | 82.1 |
| 2 | OntoSmall | 82.3 | 81.4 | 82.0 | 77.0 |
| 3 | PropSmall | - | - | 87.0 | 79.1 |

**Table 3.4.** SRL accuracy given gold syntax and pre-identified predicate-argument pairs under various train/test conditions. Rows correspond to the dataset from which the train data was used. Columns identify the labelset and the data from which the test and dev sets were used.

| | | OntoFull | | PropSmall | |
|---|---|---|---|---|---|
| | **train** | ArgN | SFT | ArgN | SFT |
| 0 | OntoFull | 88.5 | 87.9 | 86.3 | 85.8 |
| 1 | OntoMed | 87.7 | 87.0 | 85.2 | 85.5 |
| 2 | OntoSmall | 82.8 | 81.7 | 81.1 | 78.7 |
| 3 | PropSmall | - | - | 87.3 | 82.1 |

**Table 3.5.** Dev results corresponding to Table 3.4.

### 3.4.3 SRL

Table 3.2 shows that our SRL model performs well compared to published work on the English CoNLL-2009 task using gold dependencies and part-of-speech tags. It also shows the baseline performance on the SRL task I use in the remainder of the chapter (i.e. gold predicate-argument pairs are pre-identified and predicate sense is not evaluated). I include the two baselines from the literature of which I am aware that use gold syntax for English CoNLL-2009.

Table 3.4 provides insights into the PropSmall SRL data and contrasts the ArgN and SFT labelsets. Unsurprisingly, regardless of the labelset, our SRL models perform worse when fewer training examples are available. When train and test are both from a random sample of Ontonotes (i.e. the OntoFull columns) the degradation as a function of training size is

| | | **syntax=gold** | | **syntax=none** | |
|---|---|---|---|---|---|
| | **setting** | ArgN | SFT | ArgN | SFT |
| 0 | SRL | 87.0 | 79.1 | 82.7 | 79.1 |
| 1 | SRL\|SPRL | 87.7 | 80.2 | 83.2 | 80.4 |
| 2 | SRL\|SPRL$^\star$ | 86.8 | 80.5 | 84.5 | 79.4 |
| 3 | SRL+SPRL | 86.5 | 80.7 | 84.4 | 78.4 |
| 4 | SRL+SPRL$^\star$ | 86.3 | 79.8 | 83.8 | 78.1 |

**Table 3.6.** Accuracy of SRL argument labeling in isolation, given SPRL, or modeled jointly with SPRL; $^\star$ indicates second-order SPRL features.

| | | **syntax=gold** | | **syntax=none** | |
|---|---|---|---|---|---|
| | **setting** | ArgN | SFT | ArgN | SFT |
| 0 | SRL | 87.3 | 82.1 | 84.2 | 79.7 |
| 1 | SRL\|SPRL | 87.6 | 82.4 | 85.2 | 81.1 |
| 2 | SRL\|SPRL$^\star$ | 87.1 | 82.4 | 84.7 | 80.9 |
| 3 | SRL+SPRL | 86.6 | 81.7 | 83.2 | 79.3 |
| 4 | SRL+SPRL$^\star$ | 86.3 | 81.4 | 82.6 | 79.3 |

**Table 3.7.** Dev results corresponding to Table 3.6.

roughly independent of the tagset. However, training on the random subsets and testing on PropSmall hurts SFT prediction ($> 4.4$ decrease) more than ArgN ($< 2.3$ decrease). Rows 2 and 3 show a large contrast between ArgN and SFT prediction on the two datasets.

### 3.4.4 SRL using SPRL

Table 3.6 shows the SRL results on test data from models that incorporate varying amounts of SPRL information. The SRL model uses no SPRL annotations, SRL|SPRL and SRL|SPRL$^\star$ use gold annotations at test time, while SRL+SPRL and SRL+SPRL$^\star$ only use SPRL annotations at training time. Intuitively, SPRL set-valued labels provide refinements of the coarser-grained SRL labels. Comparing rows 0 and 1, we see that in all cases, features of

| | | syntax=gold | | syntax=none | |
|---|---|---|---|---|---|
| | **setting** | ArgN | SFT | ArgN | SFT |
| 0 | SPRL | 80.9 | | 80.7 | |
| 1 | SPRL* | 81.7 | | 80.8 | |
| 2 | SPRL\|SRL | 81.5 | 81.4 | 82.0 | 81.4 |
| 3 | SPRL*\|SRL | 81.8 | 80.8 | 81.7 | 81.8 |
| 4 | SRL+SPRL | 81.2 | 81.1 | 81.0 | 80.9 |
| 5 | SRL+SPRL* | 81.3 | 81.3 | 81.2 | 81.1 |

**Table 3.8.** Multi-label F1 of SPRL in isolation, given SRL, or modeled jointly with SRL.

| | | syntax=gold | | syntax=none | |
|---|---|---|---|---|---|
| | **setting** | ArgN | SFT | ArgN | SFT |
| 0 | SPRL | 79.3 | | 78.9 | |
| 1 | SPRL* | 79.5 | | 79.1 | |
| 2 | SPRL\|SRL | 79.4 | 79.6 | 79.2 | 79.3 |
| 3 | SPRL*\|SRL | 79.8 | 80.0 | 79.7 | 79.6 |
| 4 | SRL+SPRL | 79.2 | 79.3 | 78.9 | 78.9 |
| 5 | SRL+SPRL* | 79.6 | 79.6 | 79.1 | 79.1 |

**Table 3.9.** Dev result corresponding to Table 3.8

observed gold SPRL annotations allow us to learn better models. Our results are mixed for adding higher-order features and for jointly modeling SRL with SPRL. Comparing row 0 to rows 3-4, we see inferred SPRL helping SFT labeling when gold syntax is available and helping ArgN labeling when syntax is not available.[10]

### 3.4.5 SPRL with SRL

Table 3.8 shows results for SPRL evaluated as a retrieval task with F1. The results of these models are much more invariant to the availability of our syntactic features than were the SRL results of Table 3.6. The models with second-order property factors in row 1

[10] Note that in the dev results of Table 3.7, the row 1 models excel those of row 0 by even larger margins than on test while rows 3 and 4 actually perform worse than those of row 0.

improve over those without in row 0. Conditioning on gold SRL or jointly modeling SRL and SPRL generally helps except in some cases where the second-order property factors are present.

### 3.4.6 SPRL Baselines

To the best of our knowledge, the work of Reisinger et al. (2015) contains the only prior SPRL result and, according to personal correspondence with some of the authors, their predictive models were not a primary goal of that work. A key contribution of this chapter is that I refine the evaluation and propose a model that substantially outperforms the previously evaluated models. I have modified the dataset split so as to be amenable to joint modeling at the sentence (or even the section) level which makes the prediction results released with the dataset (Reisinger et al., 2015) not directly comparable to ours.[11] Therefore, Table 3.10 replicates the approach of Reisinger et al. (2015) using our evaluation and includes two other SPRL baselines (compare to Tables 3.8 and 3.12; e.g. 71.0 versus 81.7 F1 from our model). Our re-implementation of the "Full" method in Reisinger et al. (2015) uses LibLinear (Fan et al., 2008) to fit a linear model with a property-specific bias, a feature encoding the distance and direction from the predicate to the argument and an embedding of the predicate. I tuned a property-specific regularization coefficient on dev aggregate F1.[12] Picking property-specific regularization for the non-decomposable F1 objective was tricky; I independently trained and evaluated single-property models on a grid of constants; this gave rise to true-positive,

[11] Specifically, as described in section 3.4.2, I partition training, development and test data according to the WSJ section boundaries, whereas the original splits allowed arguments from a single sentence to appear in e.g. the training and test data.

[12] In contrast, tuning a single regularization coefficient (as I did for our other models) resulted in worse held-out F1 which is made even worse if property-specific bias features are included.

false-negative, and false-positive counts for each model and, therefore, a pareto optimal set for each property; by incrementally building-up the non-dominated cross-product of property-specific models, (e.g. with a varient of Kung's algorithm) I efficiently arrived at a pareto set of full models with their associated aggregate statistics for which I could evaluate the micro-averaged F1, keep the best according to dev, and report the result on held-out test data. The table also includes two additional aggregate baselines that assign labels at the *type* level (i.e. each property is either predicted as always present or absent). The first assigns the majority label for each property according to the train+dev data. The second assigns a positive label to the $k$ most frequent properties and then optimizes $k$ for F1 on train+dev ($k = 10$ being the best). After using the train and dev data to determine which properties to always predict as positive, I evaluate those predictions on the test data.

### 3.4.7 SPRL Breakdown By Property

I now take a closer look at results from our best SPRL model, SPRL$^\star$ with gold syntax (81.7 held-out F1). Table 3.12 gives a breakdown of results by property (compare to baselines in Table 3.10 and aggregate results in Table 3.8). As with the Reisinger baseline, our best performance (95.1) is for EXISTED DURING while I get less than 30.0 F1 for DESTROYED, STATIONARY and LOCATION. Clearly, I struggle most with predicting the presence of infrequent properties. This is not surprising since our micro-averaged F1 metric on which I tuned hyper-parameters encourages us to focus on the categories with the most examples.

| **Baseline** | | **F1** | **Prec** | **Rec** | | |
|---|---|---|---|---|---|---|
| property majority | | 59.1 | 70.4 | 50.9 | | |
| max type-level F1 | | 62.9 | 48.9 | 88.3 | | |
| Reisinger et al. (2015) | | 71.0 | 67.9 | 74.4 | | |
| **Reisinger et al. (2015)** | | | | | **possible** | |
| **By Property** | **CF** | **F1** | **Prec** | **Rec** | **train** | **dev** |
| instigation | + | 76.7 | 63.3 | 97.3 | 2811 | 376 |
| volition | + | 69.8 | 56.4 | 91.6 | 2728 | 350 |
| awareness | + | 68.8 | 57.4 | 85.7 | 3021 | 390 |
| sentient | 0 | 42.0 | 54.5 | 34.1 | 1856 | 244 |
| physically existed | 0 | 50.0 | 44.4 | 57.1 | 2663 | 362 |
| existed before | + | 79.5 | 67.9 | 95.9 | 4978 | 699 |
| existed during | + | 93.1 | 89.2 | 97.4 | 6566 | 879 |
| existed after | + | 82.3 | 71.1 | 97.7 | 5358 | 729 |
| created | - | 0.0 | 100.0 | 0.0 | 549 | 73 |
| destroyed | - | 17.1 | 33.3 | 11.5 | 230 | 40 |
| changed | 0 | 54.0 | 61.4 | 48.2 | 2735 | 400 |
| changed state | 0 | 54.6 | 61.3 | 49.2 | 2705 | 396 |
| changed possession | - | 0.0 | 100.0 | 0.0 | 473 | 74 |
| changed location | - | 6.6 | 66.7 | 3.4 | 575 | 55 |
| stationary | - | 13.3 | 40.0 | 8.0 | 285 | 53 |
| location | - | 0.0 | 100.0 | 0.0 | 621 | 82 |
| physical contact | - | 21.5 | 48.5 | 13.8 | 1138 | 150 |
| manipulated | + | 72.1 | 80.9 | 65.1 | 4048 | 606 |

**Table 3.10.** Aggregate multi-label SPRL results and breakdown by property using the Reisinger et al. model. **CF** is to aid visualization: + for F1 $> 66.7$, - for F1 $< 33.3$ and $0$ otherwise. The rightmost columns report the number of positive instances in the gold train and dev sections of PropSmall.

| **Baseline** | **F1** | **Prec** | **Rec** |
|---|---|---|---|
| property majority | 56.9 | 68.0 | 48.9 |
| max type-level F1 | 62.2 | 48.4 | 87.1 |
| Reisinger et al. (2015) | 69.8 | 66.7 | 73.2 |

| **Reisinger et al. (2015)** | | | | | **possible** | |
|---|---|---|---|---|---|---|
| **By Property** | **CF** | **F1** | **Prec** | **Rec** | **train** | **dev** |
| instigation | + | 77.3 | 64.7 | 95.7 | 2811 | 376 |
| volition | + | 68.9 | 56.1 | 89.1 | 2728 | 350 |
| awareness | + | 68.3 | 58.1 | 82.8 | 3021 | 390 |
| sentient | 0 | 35.2 | 59.2 | 25.0 | 1856 | 244 |
| physically existed | 0 | 50.3 | 49.9 | 50.8 | 2663 | 362 |
| existed before | + | 78.3 | 66.4 | 95.4 | 4978 | 699 |
| existed during | + | 91.0 | 85.1 | 97.8 | 6566 | 879 |
| existed after | + | 80.3 | 68.1 | 97.9 | 5358 | 729 |
| created | - | 2.7 | 50.0 | 1.4 | 549 | 73 |
| destroyed | - | 17.4 | 66.7 | 10.0 | 230 | 40 |
| changed | 0 | 54.7 | 58.4 | 51.5 | 2735 | 400 |
| changed state | 0 | 53.5 | 54.3 | 52.8 | 2705 | 396 |
| changed possession | - | 0.0 | 100.0 | 0.0 | 473 | 74 |
| changed location | - | 16.4 | 83.3 | 9.1 | 575 | 55 |
| stationary | - | 24.6 | 66.7 | 15.1 | 285 | 53 |
| location | - | 0.0 | 100.0 | 0.0 | 621 | 82 |
| physical contact | - | 32.9 | 55.6 | 23.3 | 1138 | 150 |
| manipulated | + | 74.0 | 80.6 | 68.5 | 4048 | 606 |

**Table 3.11.** Dev results corresponding to Table 3.10.

| | | | | | **possible** | |
|---|---|---|---|---|---|---|
| **Property** | **CF** | **F1** | **Prec** | **Rec** | **train** | **dev** |
| instigation | + | 85.6 | 83.1 | 88.3 | 2811 | 376 |
| volition | + | 86.4 | 84.3 | 88.5 | 2728 | 350 |
| awareness | + | 87.3 | 85.7 | 88.9 | 3021 | 390 |
| sentient | + | 85.6 | 88.1 | 83.2 | 1856 | 244 |
| physically existed | + | 76.4 | 79.3 | 73.8 | 2663 | 362 |
| existed before | + | 84.8 | 84.1 | 85.6 | 4978 | 699 |
| existed during | + | 95.1 | 93.0 | 97.2 | 6566 | 879 |
| existed after | + | 87.5 | 84.7 | 90.5 | 5358 | 729 |
| created | 0 | 44.4 | 64.9 | 33.8 | 549 | 73 |
| destroyed | - | 0.0 | 0.0 | 0.0 | 230 | 40 |
| changed | + | 67.8 | 67.5 | 68.2 | 2735 | 400 |
| changed state | 0 | 66.1 | 67.8 | 64.4 | 2705 | 396 |
| changed possession | 0 | 38.8 | 87.0 | 25.0 | 473 | 74 |
| changed location | 0 | 35.6 | 86.7 | 22.4 | 575 | 55 |
| stationary | - | 21.4 | 100.0 | 12.0 | 285 | 53 |
| location | - | 18.5 | 58.8 | 11.0 | 621 | 82 |
| physical contact | 0 | 40.7 | 62.5 | 30.2 | 1138 | 150 |
| manipulated | + | 86.0 | 85.4 | 86.6 | 4048 | 606 |
| total | | 81.7 | 83.1 | 80.3 | | |

**Table 3.12.** Breakdown of SPRL* results on held out test data with gold syntax. Compare to baselines in Table 3.10.

| | | | | | possible | |
|---|---|---|---|---|---|---|
| **Property** | **CF** | **F1** | **Prec** | **Rec** | **train** | **dev** |
| instigation | + | 83.6 | 83.2 | 84.0 | 2811 | 376 |
| volition | + | 83.9 | 82.8 | 85.1 | 2728 | 350 |
| awareness | + | 84.7 | 85.0 | 84.4 | 3021 | 390 |
| sentient | + | 80.4 | 85.6 | 75.8 | 1856 | 244 |
| physically existed | + | 75.3 | 82.3 | 69.3 | 2663 | 362 |
| existed before | + | 84.7 | 84.9 | 84.5 | 4978 | 699 |
| existed during | + | 93.7 | 89.6 | 98.2 | 6566 | 879 |
| existed after | + | 84.7 | 81.2 | 88.5 | 5358 | 729 |
| created | - | 32.7 | 48.6 | 24.7 | 549 | 73 |
| destroyed | - | 13.6 | 75.0 | 7.5 | 230 | 40 |
| changed | + | 66.7 | 65.8 | 67.8 | 2735 | 400 |
| changed state | 0 | 65.4 | 67.2 | 63.6 | 2705 | 396 |
| changed possession | 0 | 42.2 | 65.7 | 31.1 | 473 | 74 |
| changed location | - | 19.7 | 43.8 | 12.7 | 575 | 55 |
| stationary | - | 13.8 | 80.0 | 7.5 | 285 | 53 |
| location | - | 14.6 | 50.0 | 8.5 | 621 | 82 |
| physical contact | 0 | 38.7 | 59.7 | 28.7 | 1138 | 150 |
| manipulated | + | 85.6 | 83.9 | 87.5 | 4048 | 606 |
| total | | 79.5 | 81.3 | 77.8 | | |

**Table 3.13.** Dev results corresponding to Table 3.12.

### 3.4.8 SPRL Examples

Figure 3.4 shows a variety of cherry-picked outputs from the model on dev examples. In (a) it is unclear whether **two Boston sales representatives** should actually be considered the location of the event. In (b) our model infers that the **shops** did not exist until they were opened. Our output for (d) oddly misses that **relief** was CREATED despite correctly identifying that it did not EXIST BEFORE and did EXIST AFTER. In (g), it is surprising that the model correctly predicts VOLITION and AWARENESS but misses SENTIENT. This might be due to incorrect signal from also missing PHYSICALLY EXISTED. Conversely, in the *miscalculated* predicate of example (e) the annotations identify a **reporter** that is SENTIENT and has VOLITION but not AWARENESS, while the model infers that AWARENESS indeed holds. Example (h) deals with a tricky, dubious event.

**a** In August , soon after ... replaced its president , **two Boston sales representatives** *sent* customers a letter saying ...

**b** Last year , the Irish airport authority , in a joint venture with Aeroflot , *opened* **four hard-currency duty-free shops** ...

**c Enormous ice sheets** *retreated* from the face of North America , northern Europe and Asia .

**d** The notice also *grants* **relief** for certain estate-tax returns .

**e** ... **a reporter for the Reuters newswire** *miscalculated* the industrial average 's drop as a 4 % decline ...

**f She and her husband** *started* a small printing business and need the car for work as well as for weekend jaunts .

**g** In 1979 , **the pair** *split* the company in half , with Walter and his son , Sam , agreeing to operate under ...

**h Mr. Paul** denies phoning and *gloating* .

| **Property** | **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** |
|---|---|---|---|---|---|---|---|---|
| instigation | + | | | | + | + | + | + |
| volition | + | | | | + | + | + | + |
| awareness | + | | | | + | + | + | + |
| sentient | + | | | | + | + | ▨ | + |
| physically existed | + | + | ▨ | | + | + | ▨ | + |
| existed before | + | ▨ | + | | + | + | + | + |
| existed during | + | + | + | + | + | + | + | + |
| existed after | + | + | + | + | + | + | + | + |
| created | | + | | ▨ | | | | |
| destroyed | | | | | | | | |
| changed | | + | + | + | | + | + | + |
| changed state | | + | + | ▨ | | + | + | |
| changed possession | | | + | | | | | |
| changed location | | | ▨ | | | + | | |
| stationary | | ▨ | | | | | | |
| location | ▨ | | | | | | | |
| physical contact | + | + | ▨ | | | + | | |
| manipulated | | + | + | + | | | | |

**Figure 3.4.** SPRL predictions from SPRL* with gold syntax; highlighted cells reflect disagreement with annotator.

## 3.5 Related Work

The evaluation of Reisinger et al. accompanying the SPRL data release is the most closely related to our work. However, our experiments address several concerns with their setup. I split the data on section boundaries rather than randomly selecting predicate-argument pairs. I incorporate features from the SRL literature and allow properties to be predicted jointly, whereas their setup used a deliberately simple set of features and predicted properties independently. Our treatment of SPRL as multi-label classification also leads to a different evaluation metric. Table 3.10 shows the baseline for the new data splits and evaluation metric. Several authors have considered trade-offs in annotator effort and data-sparsity in arising in traditional SRL annotations (Loper, Yi, and Palmer, 2007; Yi, Loper, and Palmer, 2007; Zapirain, Agirre, and L. Màrquez, 2008).

## 3.6 Conclusions and Future Work

I established the best reported results for SPRL under a simple multi-label classification paradigm when predicate-argument pairs have already been identified. I sought improvements to our SPRL model by including pairwise proto-role factors and factors that join categorical role variables with proto-role variables. I also investigated the contrast between ArgN and semantic function tags as the underlying theory for categorical role labeling and I looked at the importance of dependency parse information to the model.

Surprisingly, I find that observed syntax and semantic roles give little boost to SPRL F1 (at most, 1.3 absolute) and that SFT SRL prediction also gains relatively little from using

SPRL or syntax. These negative results deserve further investigation. I believe that future work into improved joint models should show stronger interactions between SRL and SPRL. In contrast, our best ArgN SRL model on the same predicate-argument instances makes large gains of 4.5 absolute F1 over the syntax-free analog and 0.7 over the analog without SPRL. Furthermore, when syntax is not available, ArgN SRL benefits from having SPRL annotations available at training time, improving by 1.7 absolute held-out F1.

In Chapter 5 I explore ways to better leverage the ordinal nature of the collected responses and Chapter 7 points to modeling SPR2.x data (White et al., 2016) which includes multiple, overlapping annotators.

Figure 3.5 shows the feature templates used for unary factors on SRL role and SPRL property variables.[13] For experiments with no syntax, syntactic features are filtered from this list. Pairwise factors in this chapter include only an identity feature for each possible configuration. Also, two of our experiments predict the verb sense which variables reference a single index in the sentence rather than a pair of indexes as with predicate argument pairs. I obtained our set of sense templates that only look at a single token by removing any features in Figure 3.5 that look at multiple tokens (include pathGrams and sentlen features) and doing string substitution to replace *(c)* with *(p)* resulting in 85 unique templates.

[13] For details of the template language, see `https://github.com/mgormley/pacaya/blob/master/src/main/java/edu/jhu/featurize/TemplateLanguage.java`

bc0(-1(c))
bc0(-1(p))
bc0(1(c))
bc0(1(p))
bc0(c)
bc0(lmc(c))
bc0(lmc(p))
bc0(lnc(c))
bc0(lnc(p))
bc0(lns(c))
bc0(lns(p))
bc0(p)
bc0(rmc(c))
bc0(rmc(p))
bc0(rnc(c))
bc0(rnc(p))
bc0(rns(c))
bc0(rns(p))
bc1(c)
bc1(p)
chpre5(c)
chpre5(p)
deprel(c)
deprel(p)
lc(c)
lc(p)
bc0(head(c))
bc0(head(p))
bc0(highsn(c))
bc0(highsn(p))
bc0(highsv(c))
bc0(highsv(p))
bc0(lowsn(c))
bc0(lowsn(p))
bc0(lowsv(c))
bc0(lowsv(p))
capitalized(c)
capitalized(p)
deprel(-1(c))
deprel(-1(p))
deprel(1(c))
deprel(1(p))
deprel(lmc(c))
deprel(lmc(p))
deprel(lnc(c))
deprel(lnc(p))
deprel(lns(c))
deprel(lns(p))
deprel(rmc(c))
deprel(rmc(p))
deprel(rnc(c))
deprel(rnc(p))
bc0(dir(seq(path(p,c))))
bc0(seq(children(p)))
bc0(seq(line(p,c)))
bc0(seq(path(p,c)))
continuity(path(p,c))
deprel(bag(children(p)))
deprel(dir(seq(path(p,c))))
deprel(head(c))
deprel(head(p))
deprel(highsn(c))
deprel(highsn(p))
deprel(highsv(c))
deprel(highsv(p))
deprel(lmc(c))+word(c)
deprel(lowsn(c))
deprel(lowsn(p))
deprel(lowsv(c))
deprel(lowsv(p))
deprel(rmc(c))+word(c)
deprel(rnc(c))+word(c)
deprel(seq(children(p)))
deprel(seq(line(p,c)))
deprel(seq(path(p,c)))
distance(p,c)+relative(p,c)
lemma(bag(path(c,lca(p,c))))
lemma(bag(path(p,c)))

**Figure 3.5.** Unary feature templates for roles and proto-role properties (Table 1 of 2).

lemma(1(c))
lemma(c)
lemma(p)
pathGrams
pos(-1(c))
pos(-1(p))
pos(1(c))
pos(1(p))
pos(c)
pos(lmc(c))
pos(lmc(p))
pos(lnc(c))
pos(lnc(p))
pos(lns(c))
pos(lns(p))
pos(p)
pos(rmc(c))
pos(rmc(p))
pos(rnc(c))
pos(rnc(p))
pos(rns(c))
pos(rns(p))
sentlen
word(-1(c))
word(c)
word(p)
wordTopN(c)
wordTopN(p)

deprel(rns(c))
deprel(rns(p))
distance(p,c)
geneology(p,c)
lemma(-1(p))
lemma(head(c))
lemma(lmc(c))
lemma(rmc(c))
len(path(p,c))
pos(head(c))
pos(head(p))
pos(highsn(c))
pos(highsn(p))
pos(highsv(c))
pos(highsv(p))
pos(lowsn(c))
pos(lowsn(p))
pos(lowsv(c))
pos(lowsv(p))
pos(p)+pos(c)
pos(p)+word(c)
relative(p,c)
word(head(p))
word(lmc(c))
word(lns(c))
word(p)+pos(c)
word(rmc(c))
word(rns(c))

lemma(c)+lemma(p)
lemma(c)+word(head(c))
lemma(lowsv(c))
lemma(p)+lemma(1(p))
lemma(seq(line(p,c)))
lemma(seq(path(c,lca(p,c))))
lemma(seq(path(p,c)))
pos(-1(p))+pos(p)
pos(1(c))+pos(c)
pos(bag(children(p)))
pos(bag(noFarChildren(c)))+word(rmc(c))
pos(c)+deprel(bag(children(c)))
pos(c)+distance(p,c)+relative(p,c)
pos(dir(seq(path(p,c))))
pos(p)+deprel(bag(children(p)))
pos(p)+distance(p,c)+relative(p,c)
pos(p)+pos(c)+distance(p,c)
pos(p)+pos(c)+distance(p,c)+relative(p,c)
pos(p)+pos(c)+relative(p,c)
pos(seq(children(p)))
pos(seq(line(p,c)))
pos(seq(path(p,c)))
word(bag(children(p)))
word(c)+pos(seq(children(c)))
word(c)+word(1(c))
word(p)+deprel(bag(children(p)))
word(p)+word(c)
word(p)+word(c)+pos(p)+pos(c)
word(seq(line(p,c)))

**Figure 3.6.** Unary feature templates for roles and proto-role properties (Table 2 of 2).

# Chapter 4

# Conditional SPRL with TorchFactors

## 4.1 Introduction

While the results in Chapter 3 established a useful introduction and benchmark for investigating computational models of SPR and showed substantial improvements over previous art, I had anticipated more significant differences between the pairwise models and the independent property models. To me, the relationships between many groups of properties seem intuitive and clear; so why did the independent logistic regression model appear so competitive to the loopy CRF? And if improved features are so much more important than more faithful models, is there even a reason to consider conditional probabilistic models over recent neural approaches? Unless a case can be made for a probabilistic CRF model, many of the questions explored in subsequent chapters of this thesis may be practically irrelevant for this domain.

Upon further reflection, I wondered if the experiments of Chapter 3 were vulnerable to a number of confounding issues arising from the following:

1. a feature set and training regime that did not match most contemporary systems

2. the use of approximate inference and the use of an improper scoring rule for hyper-parameter optimization

3. an inadequate evaluation metric based on independent predictions which may fail to demonstrate the value of structure in the model—especially in the face of high quality feature representations

This chapter specifically addresses these concerns with the following contributions:

1. In Section 4.2, I release a python modeling framework (TorchFactors: †x) within the pytorch ecosystem to more easily leverage ongoing advances in neural representations and optimization *within* a probabilistic CRF paradigm.

2. Within the new framework, I revisit SPR experiments with exact inference on a subset of the SPR properties and use conditional log-likelihood as a proper scoring rule for hyper-parameter optimization.

3. In Section 4.4, I introduce the benchmark task of *Semantic Conditional Proto-Role Labeling* (SCPRL) which evaluates models on their ability to identify properties *given* observed labels for a subset of other properties. The subsets to condition on will be given only at test-time.

## 4.2 TorchFactors

I first turn to describing the motivation for and design choices of my pytorch-based CRF framework: *torchfactors*.

### 4.2.1 Motivation

In follow-up work to my experiments in Chapter 3, I have contemplated a variety of possible extensions including replacing binary models with nominal and ordinal models, allowing models to reason about whether each property is applicable, jointly identifying argument and predicate spans, and incorporating information about annotators. In a number of preliminary experiments, some of these model enhancements seemed to show surprisingly little impact, while choices of hyperparameters and features appeared to make large differences in prediction quality.

Meanwhile, the broader ML community had made large strides on learning frameworks in python (e.g. pytorch (Paszke et al., 2017)) that leverage state-of-the-art methods for learning, self-supervised feature extraction, regularization, automatic differentiation, gpu acceleration, hyper-parameter tuning, experiment logging and visualization. [1] Unfortunately, many of these practical advances were inaccessible from the java-based Pacaya learning framework used in Chapter 3. Because of my background contributing to reinforcement-learning for prioritizing structured prediction search (J. Jiang, A. R. Teichert, et al., 2012; J. Jiang, A. Teichert, et al., 2012), subsequent efforts prioritizing message passing for approximate inference via belief propagation and my contributions to work investigating

[1] See also flair (Akbik, Blythe, and Vollgraf, 2018; Akbik et al., 2019), scipy/numpy (Jones, Oliphant, Peterson, et al., 2001; van der Walt, Colbert, and Varoquaux, 2011), pandas (McKinney, 2010)

neural models for SPRL using PyTorch (Rudinger et al., 2018), I had already created a concise python implementation of generalized belief propagation based on tensor operations. In principle, porting the code to use pytorch tensors would allow exact and approximate models to immediately leverage neural features and to be trained via automatic differentiation with all of the tools available in the python and pytorch ecosystems. Perhaps these tools would make it easier to identify meaningful modeling insights.

### 4.2.2 Overview and Design

This section presents the result of several design iterations on the TorchFactors library.

An overarching goal was to remove unnatural coupling between the choice of what was being modeled, how it was being modeled, the method of performing inference under the model, the process for fitting model parameters, and evaluation of model performance. I wanted experiments to be able to share as much code as possible so as to avoid confounding differences in implementation across experiments—allowing real differences in performance to be exposed more clearly. As a secondary goal, I wanted the library to support type-checking and intellisense completion from within popular IDEs (in particular, VS Code) so as to help a modeler quickly catch mistakes. I also wanted batching to be optionally supported but without needing to clutter modeling code.

The TorchFactors library leverages the pytorch library (Paszke et al., 2017) and the factor-graph formalism (Frey et al., 1997) for specifying joint probabilistic distributions as globally normalized products of so-called *factors*. In code and in this text, I abbreviate with tx with the lowercase "t" for "torch" resembling the summation operator: $+$ and "x" resembling the product operator: ×. It borrows significant inspiration from 'FACTORIE' (McCallum, Schultz, and Singh, 2009).

Having a probability "Model"[2] for a particular class of objects requires determining the set of possible instances of that class by specifying the "member instance *variables*"[3] and then specifying a non-negative, real-valued *potential* for each each possible instantiation of the class. The probability of an instance with a particular *assignment* to (or *configuration* of) its member variables is defined to be proportional to its potential—that is, it is equal to the potential of the assignment divided by the sum (or integral) of all assignment potentials. The computation of this integral as a function of a particular class is known as the *partition function*. A convenient way to specify the potential function is as a product of local potential functions that each look at only some subset of the member variables. In principle, these can be parameterized, and the parameters of the entire model can be trained to maximize the likelihood of observed objects under the model.

Often, some member variables can be assumed to be *observed* even at test time (in contrast to *annotated* variables whose values are known for training instances but not at test time and *latent* variables whose true values are never known). In such cases, one can consider a *Conditional Model* where, as before, the probability of an assignment is proportional to the potential for that assignment but where the partition function doesn't integrate over those always-observed variables.

[2] It is common to overload the term "model" to both signify a family of distributions with parameters to be tuned or a specific setting of those parameters (and therefore a specific distribution over objects). If I ever need to disambiguate, I will use the term "model class" to indicate the former and "trained model" or "model instance" to indicate the latter.

[3] The reuse of terminology from object-oriented programming is deliberate.

### Subjects

I use the term *Subject* to represent the thing being modeled. It is common to describe the probability of the label given the input example as $P(y = Y|x = X)$. A subject class represents the concatenation of $Y$ and $X$, and an instance of the class represents the concatenation of $x$ and $y$ as well as the corresponding type of each component variable (see *usage* below). The tx framework requires the researcher to describe the data being modeled by creating a python class that inherits from the tx *Subject* class. This allows various modeling approaches and evaluations to refer to the same underlying Subject class. Python type hints are used to allow both models and evaluations to be aware of the data types of the information available for a particular subject type, facilitating better linting, static type checking, and IDE intellisense. Stacking of multiple instances into a single, batched, Subject is handled automatically so that inference on batches is done identically to inference on a single instance.

### Variables, Domains, and Usage

As mentioned, the probability of a subject instance is a function of the values currently assigned to its instance variables. To support easy definitions of models on large collections of variables and to obviate batching considerations when defining a model, tx uses tensor variables where each cell of the tensor corresponds to a separate variable while the tensor variable supports slicing to produce smaller subsets of variables. A subject class can define any number of tensor variables using instances of the `VarField` class. Each variable has an associated *domain* (the possible values that the variable can be assigned) which applies to all cells of the tensor variable, an associated tensor holding the current "value" from the domain for each cell, and an additional tensor holding a corresponding specification of the so-called *usage* for the variable represented by that cell. The "usage" for each cell of each variable takes one of the following values and impacts how the current value assigned to the variable impacts potential functions:

1. *observed* or *clamped*: Any potential function that depends on the value of this variable should be overridden to return 1 if the variable is assigned the current value, and 0 otherwise.

2. *padding*: Any potential function that depends on the value of this variable should be overridden to always return 1. In other words, the value of this variable should be ignored.

3. *latent* or *annotated*: Potential functions depending on the value of this variable are not overridden.

Note that during inference, "annotated" and "clamped" usages behave identically to "latent" and "observed" respectively, but the availability of the former pair allows for *temporarily* toggling whether or not a variable should be treated as "latent" or "observed". This facilitates computing the log-likelihood of the data—the difference between the partition function when annotated variables are clamped vs when they are free.

Figure 4.1 shows a snippet of python code defining a derived Subject class for SPRL. There will be some number of features and some number of properties associated with each subject instance. Other, non-crf-variable information (like the names corresponding to the list of ratings) can also be included.

```
from dataclasses import dataclass

import torchfactors as tx


@tx.dataclass
class SPRL(tx.Subject):
    features: tx.Var =tx.VarField(tx.OBSERVED)
    rating: tx.Var =tx.VarField(tx.ANNOTATED)
    property_names: tuple[str, ...] =()
```

**Figure 4.1.** Example tx Subject

### 4.2.3 Systems, Models and Factors

Based on a concrete Subject subclass, a tx *System* knows how to answer queries about a particular instance of that class. It is composed of a *Model* and an *Inferencer* and supports the following types of queries:

1. Product Marginals: The system's model and inferencer are used to compute the product marginals for specified subsets of variables. A query for an empty subset of variables corresponds to a request for the partition function which can be used to compute the loglikelihood of the subject conditioned on any observed variables.

2. Prediction: Given an instance of the subject class, returns a copy of the subject with values replaced to match the predictions of the system's model using the system's method of inference.

#### Factors

A factor is associated with a set of variables (parts of a subject) and represents a (possibly sparse) value for each possible assignment to that set of variables. For example, a LinearFactor in tx computes each value in the log potential function as a affine function of the input features to the factor. Models keep track of parameters and submodules which are organized into hierarchical namespaces to allow sharing of parameters while avoiding unintended sharing of parameters. Factors are like miniature systems that need to know how to answer product marginal queries involving itself and a list of other factors. The default factor implementation answers queries in a brute-force, memory-intensive way where all incoming factors and the host factor are expanded into the same dimensions and and then multiplied to form a product tensor which is then marginalized out to respond to queries. These operations are all done in log-space to avoid numeric underflow. See Gormley (2015b) and contained references for interesting work regarding structured factors and the python libraries *semiring-einsum* and *opt-einsum* for practical tools for memory- and runtime-efficient execution of queries.

### Model

A †x Model defines a probability distribution over the unobserved variables of a subject given the observed variables and any other inputs. It is defined by implementing a method that generates factors for a given subject.

Figure 4.2 shows a possible model for the SPRL subject. Note that the model depends on a particular Subject type but a variety of models can be made for the same subject type. The *factors* specifies what factors should be included given an input subject. The model object also stores all parameters necessary for creating factors. Individual factors may depend directly on input tensors provided by the subject, or they may use parameterized torch modules (arbitrary neural networks) to first transform any of those inputs prior to passing them in as features to a factor. A model is allowed to define additional latent variables that

```
from dataclasses import dataclass

import torchfactors as tx
class SPRLModel(tx.Model[SPRL]):

   def __init__(self, pairs: tuple[tuple[int, int], ...] | str,
            unary_features=True, pairwise_features=True,
            unary_bias=True, pairwise_bias=True):
      super().__init__()
      self.pairs =pairs
      self.has_unary_features =unary_features
      self.has_pairwise_features =pairwise_features
      self.has_unary_bias =unary_bias
      self.has_pairwise_bias =unary_bias

   def factors(self, x: SPRL):
      unary_features =pairwise_features =None
      if self.has_unary_features:
         unary_features =x.features.tensor
      if self.has_pairwise_features:
         pairwise_features =x.features.tensor

      num_properties =x.rating.shape[-1]
      for i in range(num_properties):
         yield tx.LinearFactor(
            self.namespace(f’var_{i}’), x.rating[..., i],
            input=unary_features, bias=self.has_unary_bias)

      if self.pairs is not None:
         pairs =self.pairs
         if pairs ==’all’:
            pairs =combinations(range(num_properties), 2)
         for subset in pairs:
            yield tx.LinearFactor(
               self.namespace(f"pair_{list(subset)}")
               *[x.rating[..., i] for i in subset],
               input=pairwise_features,
               bias=self.has_pairwise_bias)
```

**Figure 4.2.** Example tx Model

become associated with generated factors.

### CliqueModel

A CliqueModel is like a sub-model that generates factors (and possibly additional latent variables). CliqueModels, however, are designed to decoupled from any particular Subject

```
from dataclasses import dataclass

import torchfactors as tx
class SPRLModel(tx.Model[SPRL]):

   def __init__(self, clique_model, pairs: tuple[tuple[int, int], ...] |
                                        str):
      super().__init__()
      self.pairs =pairs
      self.clique_model =clique_model

   def factors(self, x: SPRL):
      num_properties =x.rating.shape[-1]
      features =x.features.tensor if self.has_features else None
      for i in range(num_properties):
         yield from self.clique_model.factors(
            x.environment, self.namespace(f’var_{i}’),
            x.rating[..., i], input=features)
      if self.pairs is not None:
         pairs =self.pairs
         if pairs ==’all’:
            pairs =combinations(range(num_properties), 2)
         for subset in pairs:
            yield from self.clique_model.factors(
               x.environment, self.namespace(
                  f"pair_{’’.join(map(str, subset))}"),
                  *[x.rating[..., i] for i in subset], input=features)
```

**Figure 4.3.** Example tx Model using a CliqueModel

```
class CollapsedProportionalOdds(CliqueModel):

   def factors(self, env: Environment, params: ParamNamespace,
            *variables: Var, input: Optional[Tensor] =None):
      # each subset of variables gets a separate weight given input
      yield make_binary_factor(
         params.namespace(’binary-configs’),
         *variables, input=input,
         minimal=True, binary_bias=False,
         get_threshold=lambda _: 1)
      # and each configuration gets a separate bias
      yield LinearFactor(
         params.namespace(’ordinal-bias’),
         *variables, input=None, bias=True, minimal=True)
```

**Figure 4.4.** Example tx CliqueModel

type so that they can be reused in various models of different Subject types. While a Model receives a Subject instance and references the names of specific variables within the Subject, CliqueModels are built over collections of variables. Models can use clique models.

Factors deal with specific grounded variables in a particular subject, but there are patterns for modeling a group of variables that can be applied across multiple variable cliques in the same graph and some of those may share parameters. A clique model can introduce parameters of its own and generates factors.

Chapter 5 will describe a number of models that can be represented in tx as clique models. Figure 4.4 gives one example. Figure 4.3 uses CliqueFactors to define a generalized version of the model in Figure 4.2. Note that the model still determines what cliques to consider based on the subject-specific variables, while the clique model specifies how those cliques will be modeled with factors (and possibly latent variables).

### Environment

Since cliques can overlap, there is an optional mechanism called an *Environment* available to keep track of what factors or latent variables have already been generated for a particular Subject instance on a given generation of factors. This can make it easier for the model to add latent variables and factors without redundancy.

### Inferencers

tx supports back propagation through loopy belief propagation as approximate inference as well as the (brute-force) exact inference method used in this chapter. I will revisit approximate inference in Chapter 6.

## 4.3 Exact-Inference SPRL With TorchFactors

This section uses tx to address three aspects of the experiments from Chapter 3, by changing the feature representation, limiting the modeling task to allow exact inference, and modifying hyper-parameter selection and model evaluation.

First, my previous experiments relied on sparse features inspired by years of feature engineering for SRL. However, implementing these features can be error-prone to replicate. In contrast, successful, dense feature representations tuned via self-supervision of massive textual datasets largely avoid the need for hand-crafted feature-sets. To reduce dependence on SRL-specific feature engineering, the experiments in this chapter are based on a dense input representation available when I started the experiments. Although even better dense representations have since been proposed, nothing in the formulation precludes swapping in newer representations.

Second, my previous experiments ran at most five iterations of the belief propagation algorithm using the Pacaya modeling and learning framework for Java. Although the algorithm is exact for tree-structured graphs, many of the models I explored (including some of the best performing models) had cycles in the factor graph, so inference and gradient steps were only heuristic. While the ability to use approximate inference to train these intractable probabilistic models is an important strength of the decompositional, crf-based approach, various aspects of the experimental results suggested the need for follow-up experiments to better disentangle the effects of modeling from the impact of approximate inference. To achieve exact inference in this chapter, I limit the task to just six of the proto-role properties[4]

[4] I used the properties shown in Table 1 of Rudinger et al. (2018): 'prop-01a-instigation', 'prop-02a-volition', 'prop-03a-awareness', 'prop-05a-physically-existed', 'prop-08a-existed-after', 'prop-12a-changed-state'. It

and use a brute force inferencer that computes the partition function by enumerating over the entire space of variable configurations.

Third, based on the ideas presented in Stoyanov, Ropson, and Eisner (2011) advocating for optimization that accounts for approximations and the ultimate objective function together with my prior experience with hyperparameter optimization, I took particular care to randomly sample a variety of hyper-parameter configurations for each proposed model architecture and to use development data to select the best performing hyper-parameters with respect to our chosen evaluation criteria. The results of these experiments and my experience in general has confirmed the strong impact of hyper-parameter tuning on the quality of the results. Specifically, albeit anecdotally, (1) model performance can often be more sensitive to hyper-parameters than to the choice of the model architectures under evaluation; (2) the best-performing hyper-parameter settings can vary greatly across the model architectures being evaluated; and (3) the best-performing hyper-parameter settings are sensitive to the choice of evaluation criteria.

These issues are especially troubling in the context of decompositional models where the richer structure of the data can naturally tend toward architectures with an increased number of hyper-parameters and a larger variety of evaluation criteria to choose from. For example, an exponential number of property subsets to potentially model effectively gives rise to a hyper parameter choosing how many and which subsets to focus on, and approximate inference under such a model gives rise to additional hyper-parameters such as the number of belief propagation iterations or even parameters governing update schedules. Indeed,

it worth noting that in an earlier run of these experiments, I inadvertently included property 11–“changed” rather than property 2–“volition” which resulted in two very similar but difficult properties: “changed” and “changed-state”. During the conditional evaluation, these two properties dominated the macro F1 score and also provided near perfect signal to each other, showing even higher gains from conditioning.

in the neural SPR experiments of Rudinger et al. (2018), we revisited the choice to use micro-averaged F1 due to its over-focus on the common properties at the expense of rarely occurring properties and additionally considered correlation-based evaluations.

To better decouple experimental conclusions from my choice of evaluation criteria, the follow-up experiments in this chapter select hyper-parameters based solely on validation likelihood rather than using a more task-specific loss function. The development data does not systematically impact any aspect of training (including early stopping and hyper-parameter selection), so that more investigations and conclusions can be made on the basis of dev results while preserving test data for evaluation prior to significant publication.

The main question of this chapter whether joint CRF binary decompositional models are able to improve over independent prediction of the properties. Under the refined conditioned described above, I investigate this question by evaluating a number of models. For each model, I compute the full conditional log-likelihood of the selected properties using exact (brute-force) inference in †x—training the parameters of each model using the *Adam* optimization algorithm as implemented in pytorch. I use the PropSmall dataset and splits from Chapter 3 except that I split the train subset into two splits: trainA (0-15) and val (16-18), using the validation set to determine when to stop training and for choice of hyper-parameters. Although this leaves me fewer training examples, it allows me to essentially insulate the development data from impacting the training.

I compare the following models:

- SPRL (as defined in Chapter 3) includes property variables $\{P_{ijq}\}$ with an independent binary classifier for each conjunction of predicate-argument pair $(i, j)$ and property $q$.
- SPRL$^\star$ (as defined in Chapter 3) has the same variables as SPRL but allows for interac-

tions between pairs of properties. For each pair of properties $q$ and $r$, there is a factor between $P_{ijq}$ and $P_{ijr}$.

- SPRL$^{t+}$ is like SPRL$^{\star}$ but with only a tree-structured subset of property pairs included. Assuming only unary and pairwise binary factors, the tree-structured subset of factors that would maximize the likelihood of the trainA subset of the data can be efficiently chosen using a maximize spanning tree algorithm executed on a complete graph with nodes representing the properties, edges representing pairs of properties, and the weight of an edge being given as follows: Let $\hat{P}_{ij}(Q_i = q_i, Q_j = q_j)$ be the empirical probably that properties $i$ and $j$ receive binary labels $q_i$ and $q_j$, and let $\hat{P}_i(Q_i = q_i)$ be the empirical probability that property $i$ receive binary label $q_i$. Then the weight $e_{ij}$ between nodes $i$ and $j$ should be $\sum_{q_i,q_j} \hat{P}_{ij}(Q_i = q_i, Q_j = q_j) \log \frac{\hat{P}_{ij}(Q_i=q_i,Q_j=q_j)}{\hat{P}_i(Q_i=q_i)\hat{P}_j(Q_j=q_j)}$.[5]
- SPRL$^{t-}$ is like SPRL$^{t+}$ except that the tree is a minimum spanning tree of the same graph.
- SPRL$^{\star}$0 is like SPRL$^{\star}$ but with only unary and pairwise bias terms available (it is not able to look at the input and will always predict the most common label for each property).

The trained models have the following hyper parameters:

- `weight_decay` (similar to an L2 regularizing constant)
- `lr` (learning rate—similar to a step size)

[5] To see that this is the case, consider that the MLE estimator for a tree-structured model achieves the unary and pairwise marginals of the empirical distribution. Thus, it can be reparameterized such that the unary factors exactly match the unary empirical distributions and so the pair-wise factors will exactly match the pairwise empirical distributions divided by the unary distributions corresponding to the edges end-points (so as to not double-count). The empirical pairwise distribution tells us how many of the training examples include each pairwise configuration and therefore multiply that probability by the corresponding log potentials yields the total additive contribution across the training set of that factor to the loglikelihood.

Training is carried out with the Adam algorithm using a single batch and evaluating training and validation log-likelihood after each epoch. “Early stopping” is performed as follows both to serve as regularization and to determine how long to spend in training: a new model is saved each time the training and validation log-likelihood are both improved by at least $0.01$ over the previously saved model, and training continues until 50 epochs pass without saving a new model.

To maximize the chance that I get good settings of hyper-parameters, I use the following two-phased, stratified approach:

1. Phase 1: For each parameter select 30 log-linearly spaced settings from $10^{-4}$ to $10^{4}$, rounding to the nearest 2 significant digits. Combine the resulting settings into 30 configurations by randomly choosing (without replacement)[6] one of the unused settings for each parameter. Train each model type under comparison (i.e. $\texttt{SPRL}$, $\texttt{SPRL}^{\star}$, $\texttt{SPRL}^{t+}$, $\texttt{SPRL}^{t-}$) on each of the 30 configurations.

2. Phase 2: For each model type under comparison and for each of the 30 original runs, reduce the range for the hyper parameter to include the largest and smallest of the three best performing settings under Phase 1 according to validation likelihood as well as the next smaller and next larger setting (if applicable). Then repeat Phase 1 on the model-type specific narrowed ranges. Evaluate the winning model from the refined sweep for each model type with respect to evaluation metrics of interest. Optionally evaluate the best model on test data for comparison to previous work.

Figures 4.1 and 4.2 summarize the results on validation and development data respec-

[6] I.e., shuffle the lists.

| | | | **averaging=micro** | | | **averaging=macro** | | |
|---|---|---|---|---|---|---|---|---|
| | **setting** | **likelihood** | F1 | P | R | F1 | P | R |
| 0 | SPRL$^{\star}$0 | 6.2 | 39.9 | 70.2 | 27.8 | 28.4 | 95.0 | 16.7 |
| 1 | SPRL | 14.3 | 82.5 | 82.8 | 82.1 | 81.9 | 82.7 | 81.1 |
| 2 | SPRL$^{t-}$ | 14.1 | 82.6 | 83.0 | 82.2 | 82.1 | 83.1 | 81.1 |
| 3 | SPRL$^{t+}$ | 18.2 | 82.4 | 83.2 | 81.7 | 81.7 | 83.0 | 80.5 |
| 4 | SPRL$^{\star}$ | 18.9 | 82.8 | 83.6 | 82.0 | 82.1 | 83.5 | 80.8 |

**Table 4.1.** Model structure comparison on six properties of val data

| | | | **averaging=micro** | | | **averaging=macro** | | |
|---|---|---|---|---|---|---|---|---|
| | **setting** | **likelihood** | F1 | P | R | F1 | P | R |
| 0 | SPRL$^{\star}$0 | 6.3 | 39.7 | 68.1 | 28.0 | 28.3 | 94.7 | 16.7 |
| 1 | SPRL | 14.0 | 81.7 | 82.1 | 81.3 | 81.3 | 82.1 | 80.6 |
| 2 | SPRL$^{t-}$ | 14.0 | 81.6 | 82.2 | 81.1 | 81.3 | 82.3 | 80.2 |
| 3 | SPRL$^{t+}$ | 18.2 | 82.0 | 83.8 | 80.3 | 81.4 | 83.7 | 79.3 |
| 4 | SPRL$^{\star}$ | 18.9 | 82.1 | 83.4 | 80.7 | 81.6 | 83.5 | 79.9 |

**Table 4.2.** Model structure comparison on six properties of dev data

tively. The likelihood column in these tables is the exponentiated average log-likelihood of each example (thus it is a number between 0 and 1, its log is proportional to the log-likelihood of the dataset under the model, and it reflects the difficulty of the dataset but not the number of examples in the dataset, so it should be relatively comparable across datasets with different numbers of examples).

A few trends are worth noting. First, even though the development data was held out during training and not use for early stopping or for hyper-parameter optimization, the likelihood results between validation and development data correspond closely. Second, for both validation and development data, the likelihood of the respective models rank as we would expect except that it is slightly surprising that SPRL$^{t-}$ does no better and possibly slightly worse on held out data than SPRL. From worst to best we have: (SPRL and SPRL$^{t-}$),

| | | | **averaging=micro** | | | **averaging=macro** | | |
|---|---|---|---|---|---|---|---|---|
| | **setting** | **likelihood** | F1 | P | R | F1 | P | R |
| 0 | SPRL$^{\star}$0 | 69.8 | 80.0 | 82.2 | 77.9 | 79.0 | 80.9 | 77.1 |
| 1 | SPRL | 72.3 | 82.5 | 82.8 | 82.1 | 81.9 | 82.7 | 81.1 |
| 2 | SPRL$^{t-}$ | 72.2 | 82.8 | 83.0 | 82.7 | 82.2 | 82.9 | 81.5 |
| 3 | SPRL$^{t+}$ | 77.0 | 85.8 | 87.1 | 84.5 | 85.6 | 87.4 | 83.8 |
| 4 | SPRL$^{\star}$ | 77.7 | 86.3 | 87.2 | 85.5 | 86.1 | 87.3 | 85.0 |

**Table 4.3.** Model structure comparison on six properties of val data conditioned on other five properties

| | | | **averaging=micro** | | | **averaging=macro** | | |
|---|---|---|---|---|---|---|---|---|
| | **setting** | **likelihood** | F1 | P | R | F1 | P | R |
| 0 | SPRL$^{\star}$0 | 70.1 | 80.5 | 83.6 | 77.6 | 79.7 | 82.6 | 76.9 |
| 1 | SPRL | 72.0 | 81.7 | 82.1 | 81.3 | 81.3 | 82.1 | 80.6 |
| 2 | SPRL$^{t-}$ | 72.1 | 82.0 | 82.0 | 82.0 | 81.7 | 82.2 | 81.1 |
| 3 | SPRL$^{t+}$ | 77.0 | 84.9 | 86.7 | 83.1 | 84.8 | 87.0 | 82.6 |
| 4 | SPRL$^{\star}$ | 77.7 | 85.3 | 87.1 | 83.7 | 85.4 | 87.4 | 83.5 |

**Table 4.4.** Model structure comparison on six properties of dev data conditioned on other five properties

SPRL$^{t+}$, SPRL$^{\star}$. Interestingly, however, SPRL$^{t-}$ achieves nearly as low likelihood as SPRL, while SPRL$^{t+}$ achieves nearly as high likelihood as SPRL$^{\star}$.

Given the large differences in likelihood, the F1 scores and other discrete metrics are notably less pronounced and consistent.

## 4.4 Semantic Conditional Proto-Role Labeling

I conclude this chapter with a novel evaluation of SPR models that seeks to highlight the value of a joint probabilistic model like the CRF models proposed in this thesis.

Without additional tuning, I evaluate the models from Section 4.3 by considering how the models allow prediction of each property when one or more of the other properties

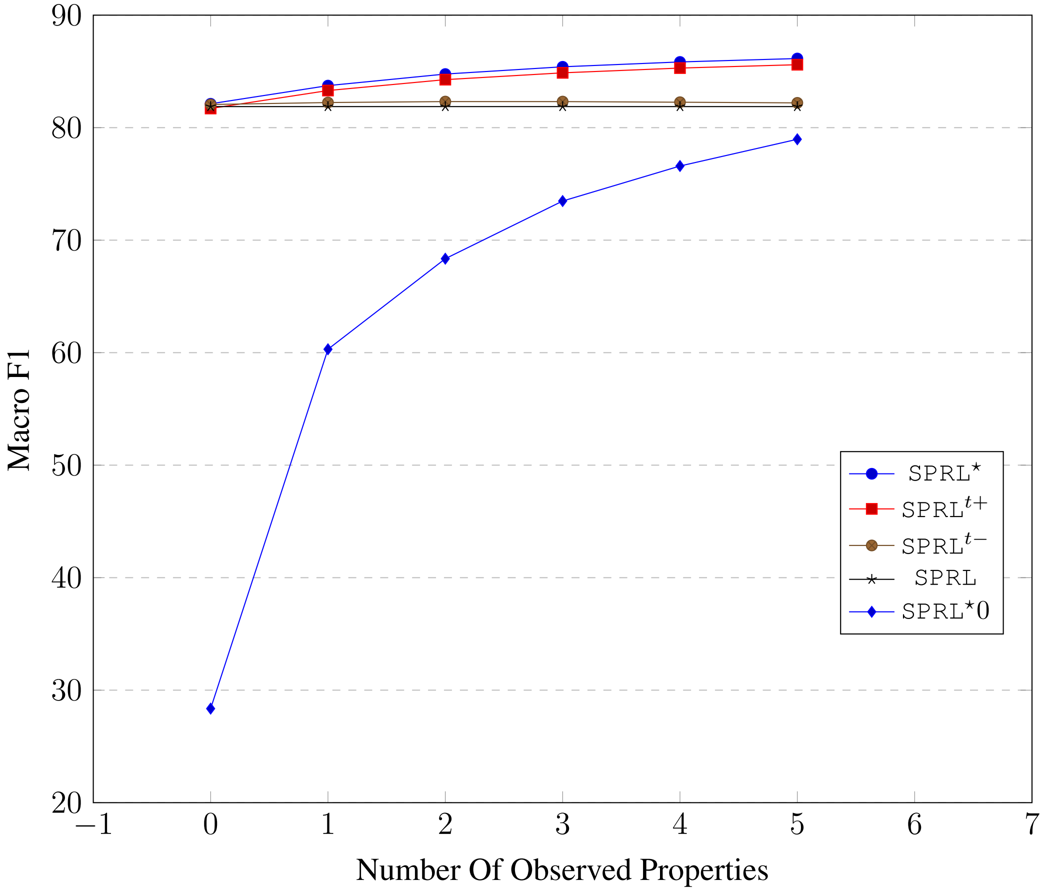


**Figure 4.5.** Conditional macro f1 on val data for six properties

are known at test time. Such a scenario might arise when extending an annotated corpus with additional properties or when using an annotation protocol that specifically supports partial annotation of only a random (or carefully chosen) subset of properties per input. For example, perhaps a project is interested in assessing 10 properties on a large corpus but only has budget to annotate 5 properties per input.[7]

Tables 4.3 and 4.4 show the performance of the models evaluating each property with the other five properties observed at test time and Figures 4.5 and 4.6 report the macro-averaged f1 score as a function of the number of properties observed at test time. Each model is

[7] Allowing deliberate- or forced- missing labels in an annotation scheme adds an interesting dimension to the question of how to make the most efficient use of an annotator's cognitive effort and is related to the question of whether to label more examples with less redundancy or fewer examples with more redundancy.

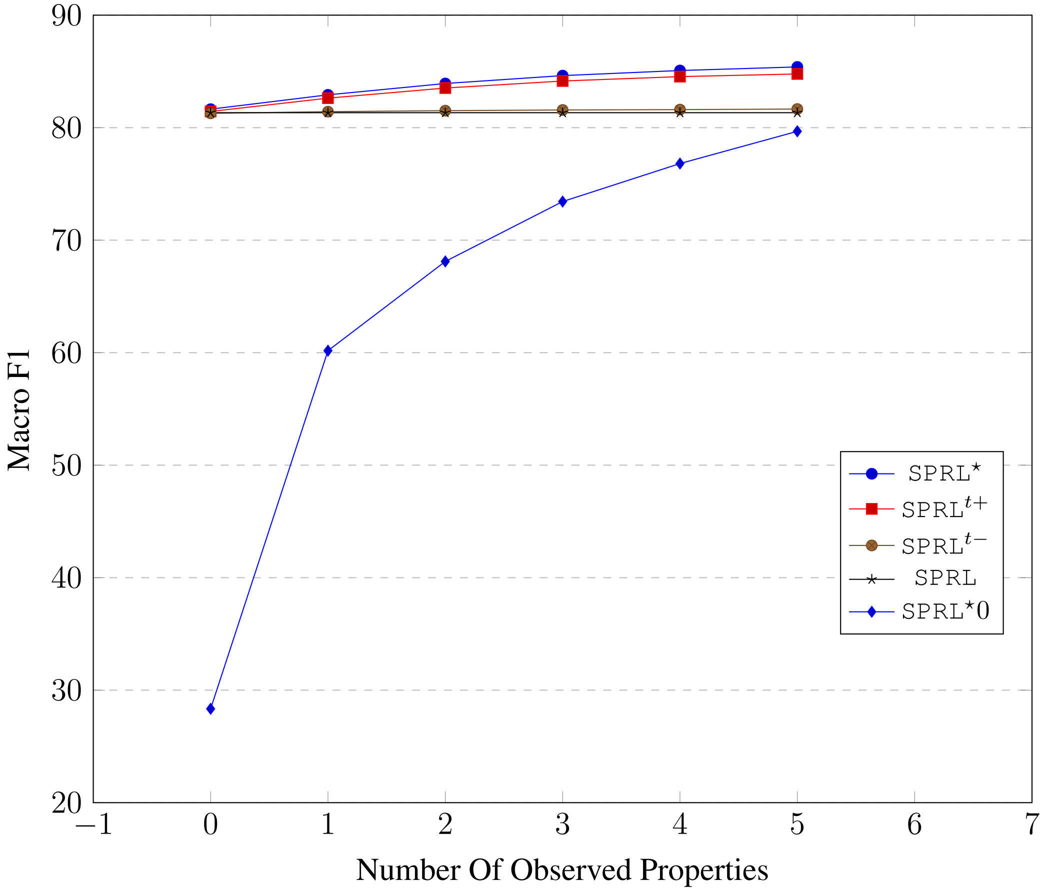


**Figure 4.6.** Conditional macro f1 on dev data for six properties

represented by a curve. The x axis shows the number of properties observed at test time (between 0 and 5). To compute the y axis, each evaluation example is duplicated once for each possible way of observing $x$ of the six properties, the macro f1 on the resulting dataset is then reported. SPRL⋆0 is a naive baseline that only has access to bias features (i.e. does not look at any features of the input). SPRL⋆0 is nearly competitive with the independent model once all but one of the properties are exposed, though there still remains a gap which indicates that features of the input are more helpful even than a perfect leave-one-out representation of the remaining five properties. Since the feature-based models are relatively close compared to SPRL⋆0, Figures 4.7 and 4.8 show the same result but exclude SPRL⋆0 from the comparison.

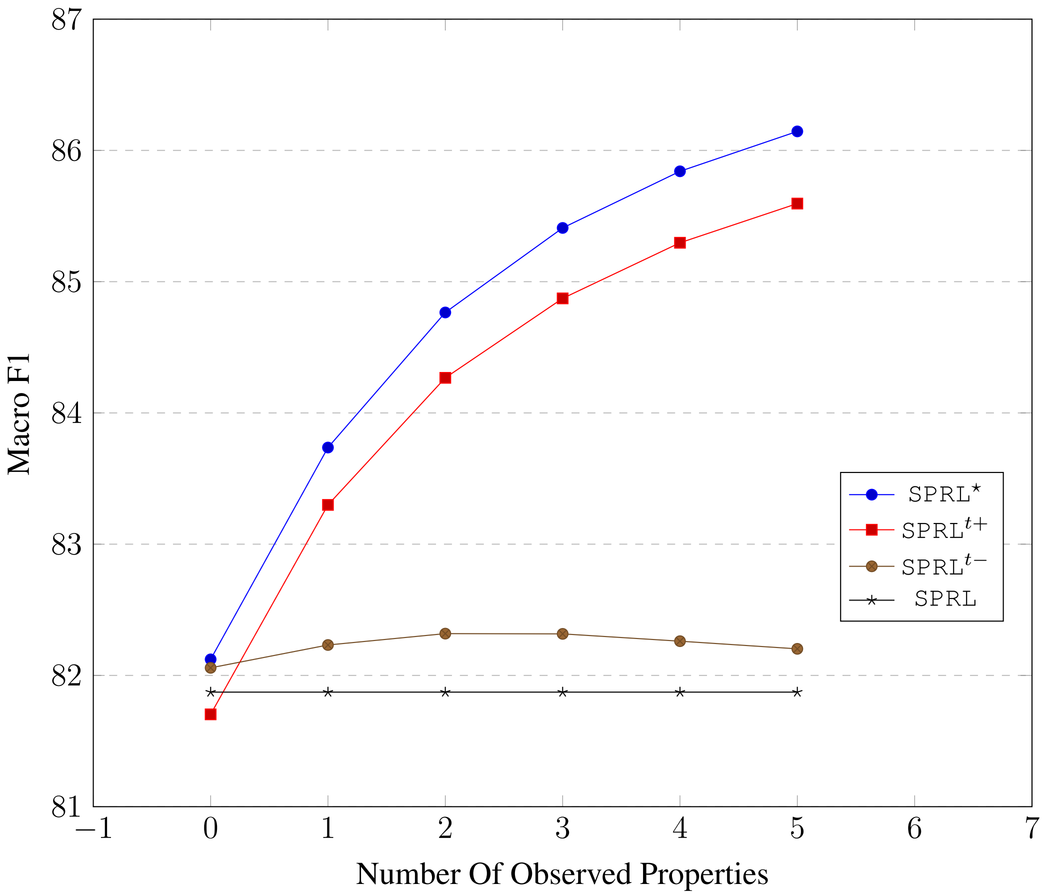


**Figure 4.7.** Conditional macro f1 on val data for six properties (excluding SPRL*0 to improve resolution)

Figures 4.11 and 4.9 show the separate performance of the six properties given various levels of observation on validation data under SPRL* and $\text{SPRL}^{t+}$ respectively (Figures 4.12 and 4.10 show the same on development data). The "existed-after", "changed-state", and "physically-existed" properties do not show much benefit in terms of F1 from any of the other properties in this subset.

Figures 4.13 and 4.14 show, respectively, the max and min tree based on the six property subset considered in this chapter (based on the training data). Figures4.15 and 4.16 show the corresponding max and min trees for the entire set of properties.

Since the max tree performs so close to the full loopy model, it is worth knowing the

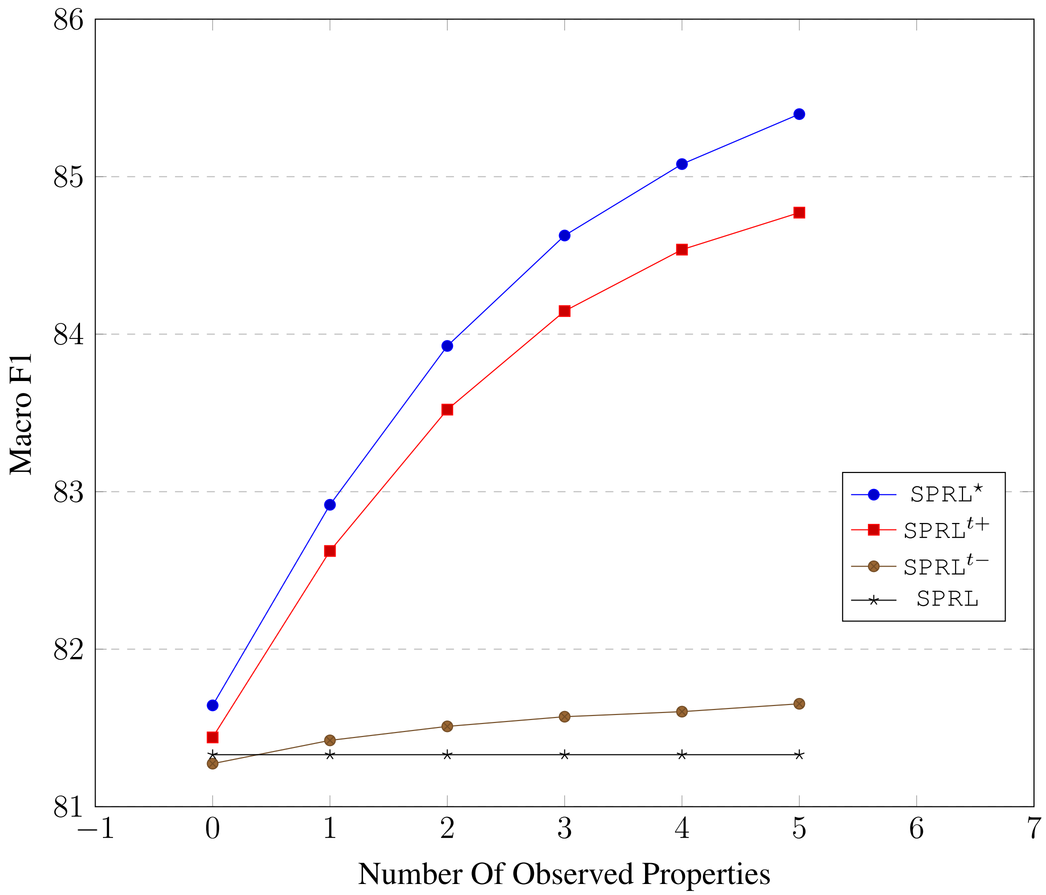


**Figure 4.8.** Conditional macro f1 on dev data for six properties (excluding SPRL⋆0 to improve resolution)

sensitivity of the tree to the the particular sample of training data.

I performed bootstrapped sampling for various sizes of samples. For each subset size, I sampled (with replacement) that number of instances from the training set and then computed the max tree based on the resulting statistics, repeating the process 1000 times and counting the distinct resulting trees. Tables 4.5 and 4.6 summarize the results for the subset of six properties and for the full set of SPR1 properties. The results suggest that the available training data is ample to reliably select a maximum tree over the six properties while the tree chosen over the entire set of properties may not be optimal with respect to another sample of data (although there will likely be much overlap as shown in the most significant edges

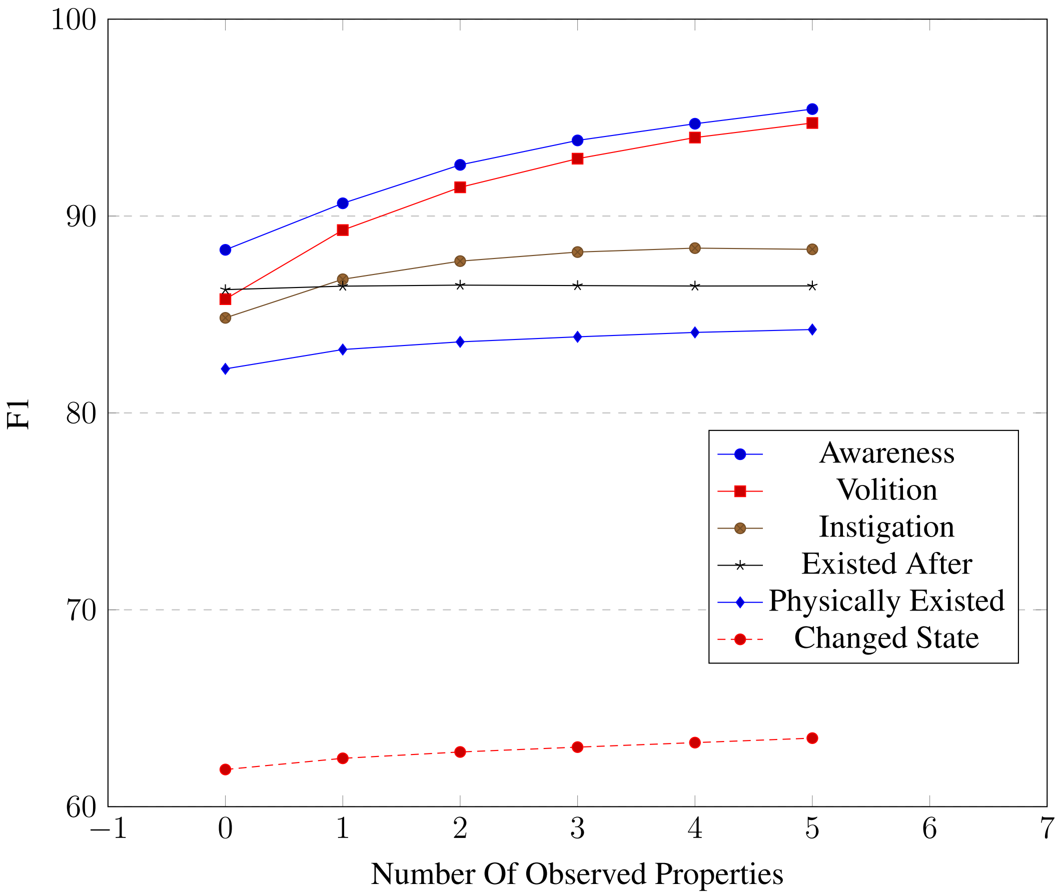


**Figure 4.9.** Property-specific conditional f1 for val data using $\texttt{SPRL}^{t+}$

selected for the most frequent trees).

## 4.4.1 Label Annotation Ordering

Since conditioning allows improved prediction of unlabeled properties, such a model might be useful in quickly building large, useful annotated corpora or in leveraging human effort at test time. This raises the question of label priority. For example, if only one label could be observed, which property should that be? If two could be observed, which two? If the second could be chosen conditioned on the first, how would our choice be different. We leave the following experiment to future work. In the conditional experiments

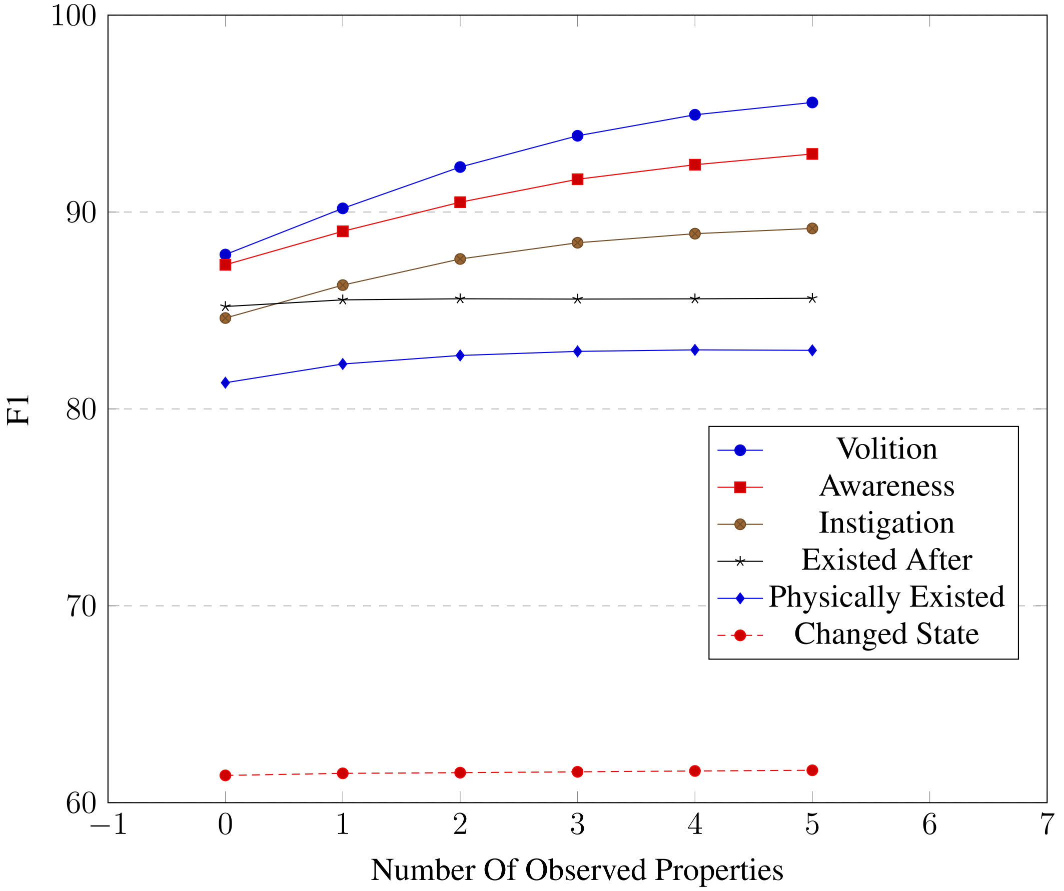


**Figure 4.10.** Property-specific conditional f1 for dev data using $\texttt{SPRL}^{t+}$

above, we evaluated on all possible subsets of the given size. Alternatively, we could have monotonically constructed a set of observed labels by adding at each size, the label that would maximize the conditional F1 of the remaining labels. So as to ensure that the choice of next label to reveal is not allowed to also select the set of properties being evaluated, properties included in the observed set should be evaluated conditioned only on the other observed properties. An example-specific order could be chosen in the same manner and such an oracle order could potentially be used a training data for a system designed to suggest the next label to obtain given features of the input and labels already obtained.

Figures 4.17 and 4.18 show the results of the second hyper-parameter pass versus the initial pass on the experiments in this chapter. The plots show that the second pass has much

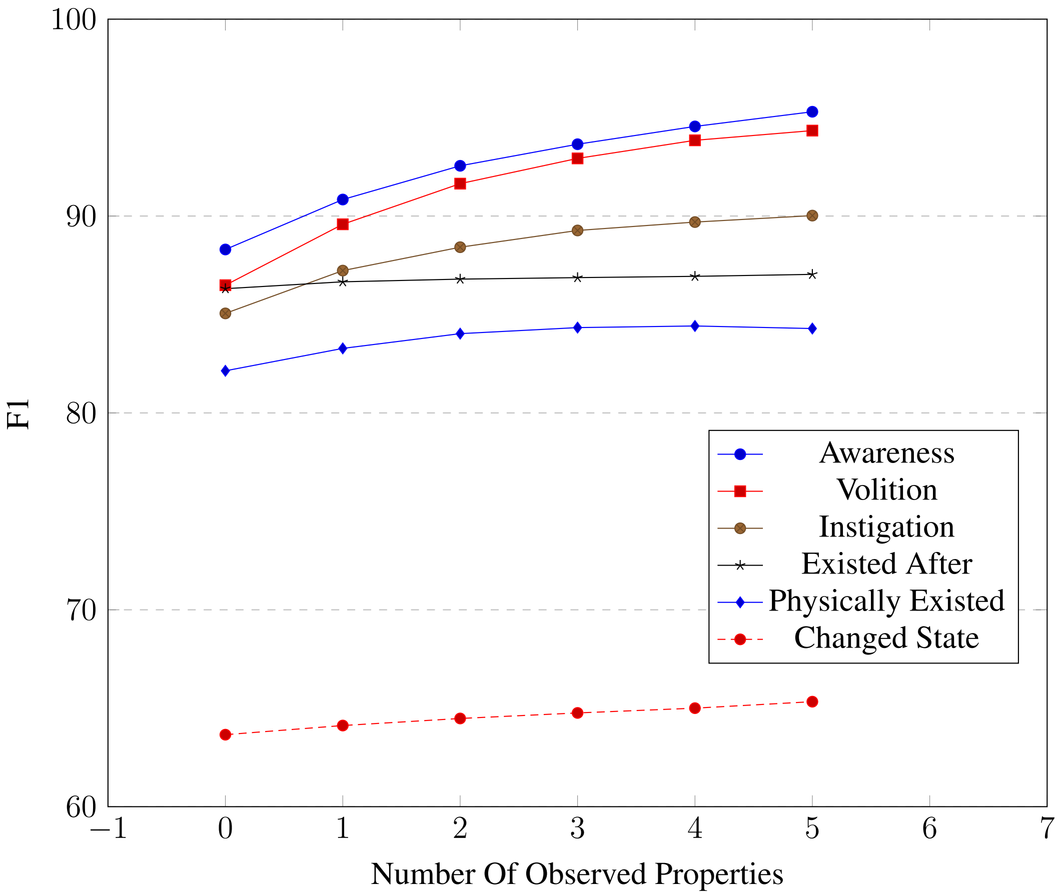


**Figure 4.11.** Property-specific conditional f1 for val data using `SPRL`*

less variation in results with respect to likelihood than the initial pass and but seldom finds a result that is much better than what was achieved during the first pass. The method of getting maximally spaced gridpoints from each dimension and then random pairing seems to be doing a good job of exploring the hyper parameter space. It would be interesting to compare the final results under the two phase paradigm against a single phase with twice as many intervals.

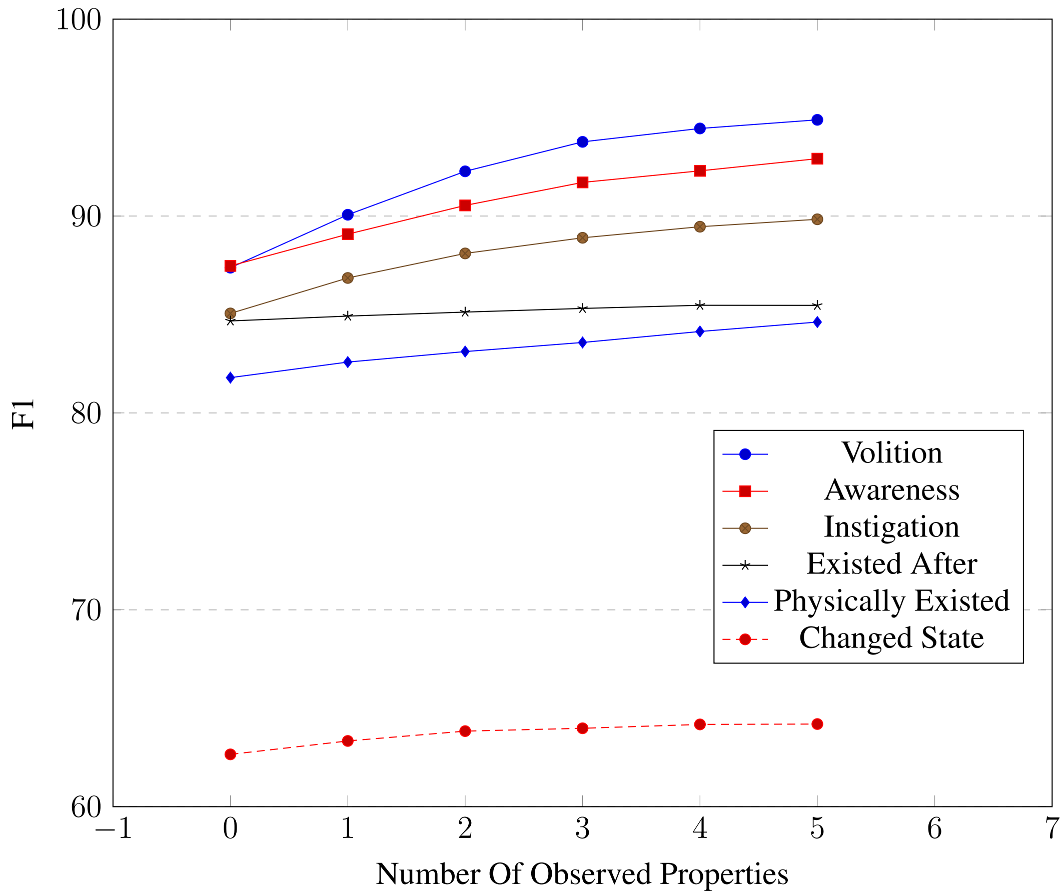


**Figure 4.12.** Property-specific conditional f1 for dev data using SPRL*

## 4.5 Conclusions

My previous SPRL experiments established baselines and benchmark evaluations for SPRL but did not satisfy me that the structured models were necessary or even helpful. This chapter removed confounding factors from the experimental design by introducing a pytorch-based modeling framework facilitating an improved experimental designing for comparing SPRL models. Specifically, the detailed experiments used exact inference and compared against well-chosen tree-structured models in addition to non-structured joint models. Finally, I introduced an SPR-related task that relies on joint knowledge to enable a single SPR model to be evaluated with respect to the models ability to transfer information

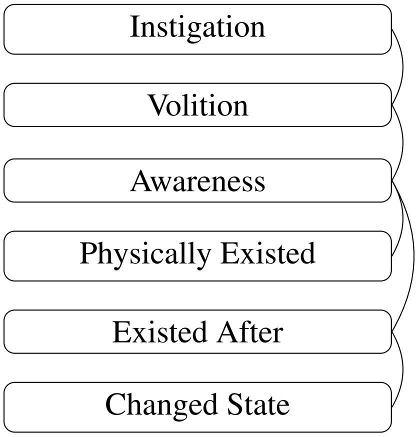


**Figure 4.13.** Max Tree on 6 Properties

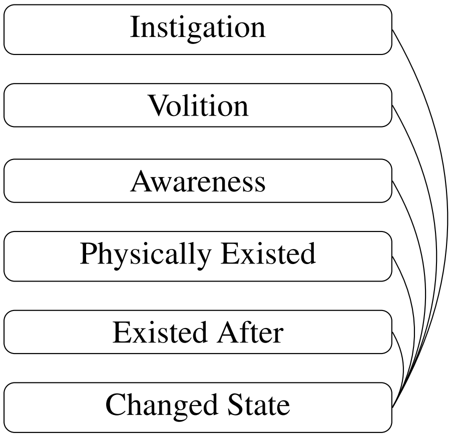


**Figure 4.14.** Min Tree on 6 Properties

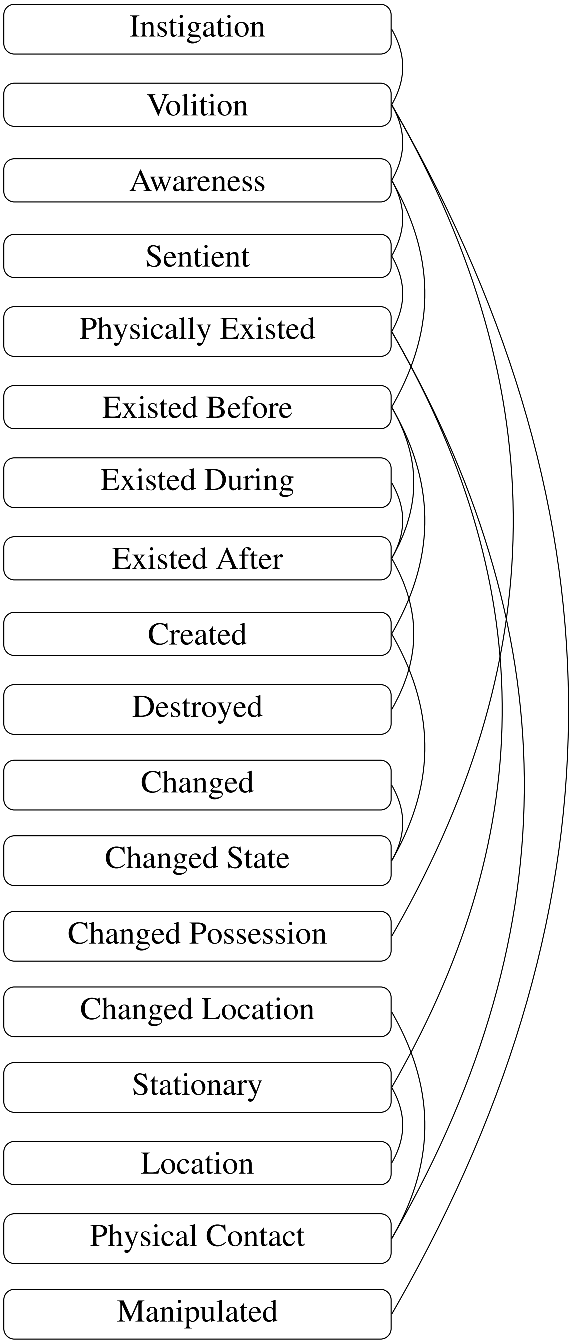


**Figure 4.15.** Max Tree on All Properties

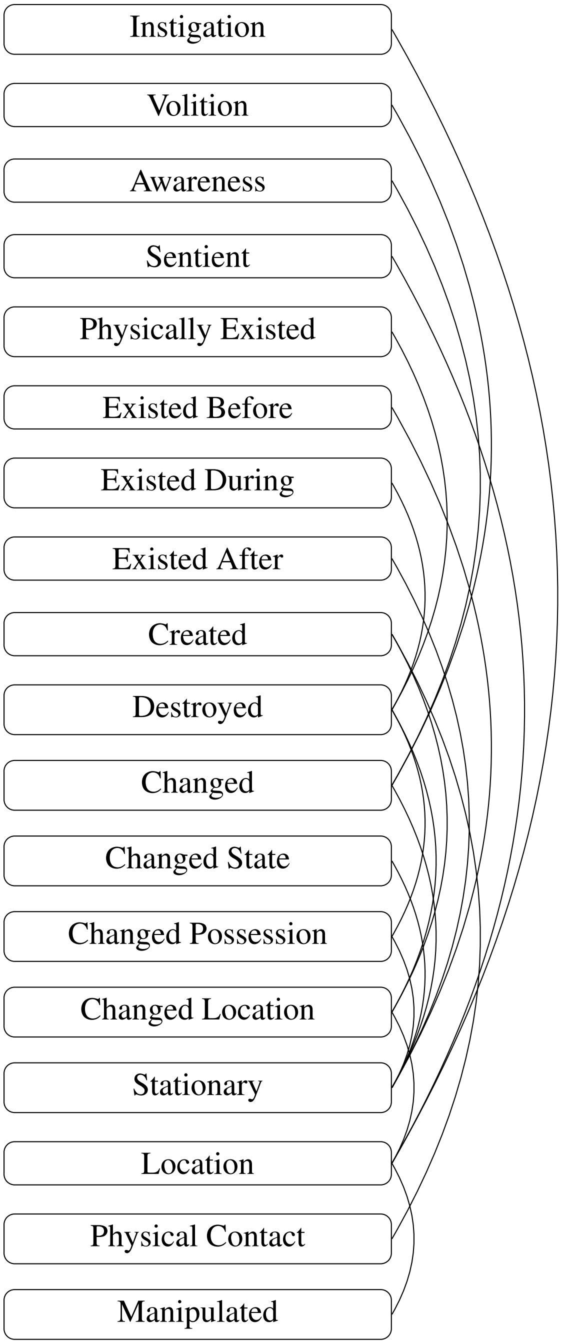


**Figure 4.16.** Min Tree on All Properties

| | Bootstrap Size | Distinct Trees | Most Frequent Tree | Relative Frequency |
|---|---:|---:|:---:|---:|
| 0 | 1 | 1 | [(0, 1), (0, 2), (0, 3), (0, 4), (0, 5)] | 100.0 |
| 1 | 4 | 102 | [(0, 1), (0, 2), (0, 3), (0, 4), (0, 5)] | 29.8 |
| 2 | 16 | 179 | [(0, 1), (1, 2), (1, 3), (1, 4), (4, 5)] | 3.5 |
| 3 | 64 | 96 | [(0, 1), (1, 2), (2, 3), (2, 4), (4, 5)] | 16.6 |
| 4 | 256 | 31 | [(0, 1), (1, 2), (2, 3), (2, 4), (4, 5)] | 47.9 |
| 5 | 1024 | 11 | [(0, 1), (1, 2), (2, 3), (2, 4), (4, 5)] | 90.2 |
| 6 | 4096 | 1 | [(0, 1), (1, 2), (2, 3), (2, 4), (4, 5)] | 100.0 |
| 7 | 16384 | 1 | [(0, 1), (1, 2), (2, 3), (2, 4), (4, 5)] | 100.0 |

**Table 4.5.** Frequencies of Max Tree based on 1000 bootstraps samples of various sizes for six properties

| | Bootstrap Size | Distinct Trees | Most Frequent Tree | Relative Frequency |
|---|---:|---:|---:|---:|
| 0 | 1 | 1 | [(0, 1), (0, 2), (0, 3), (0, 4), (0, 5), ...] | 100.0 |
| 1 | 4 | 918 | [(0, 1), (0, 2), (0, 3), (0, 4), (0, 5), ...] | 1.3 |
| 2 | 16 | 1000 | [(0, 1), (0, 2), (0, 3), (0, 5), (0, 7), ...] | 0.1 |
| 3 | 64 | 998 | [(0, 1), (0, 9), (1, 2), (2, 3), (2, 5), ...] | 0.2 |
| 4 | 256 | 744 | [(0, 1), (1, 2), (1, 17), (2, 3), (2, 5), ...] | 0.9 |
| 5 | 1024 | 208 | [(0, 1), (1, 2), (1, 17), (2, 3), (2, 5), ...] | 4.5 |
| 6 | 4096 | 70 | [(0, 1), (1, 2), (1, 17), (2, 3), (2, 5), ...] | 7.6 |
| 7 | 16384 | 36 | [(0, 1), (1, 2), (1, 12), (1, 17), (2, 3), ...] | 15.3 |

**Table 4.6.** Frequencies of Max Tree based on 1000 bootstraps samples of various sizes for all properties

from known properties to other properties of interest. The tx library is available as open-source software via https://github.com/teichert/torchfactors.

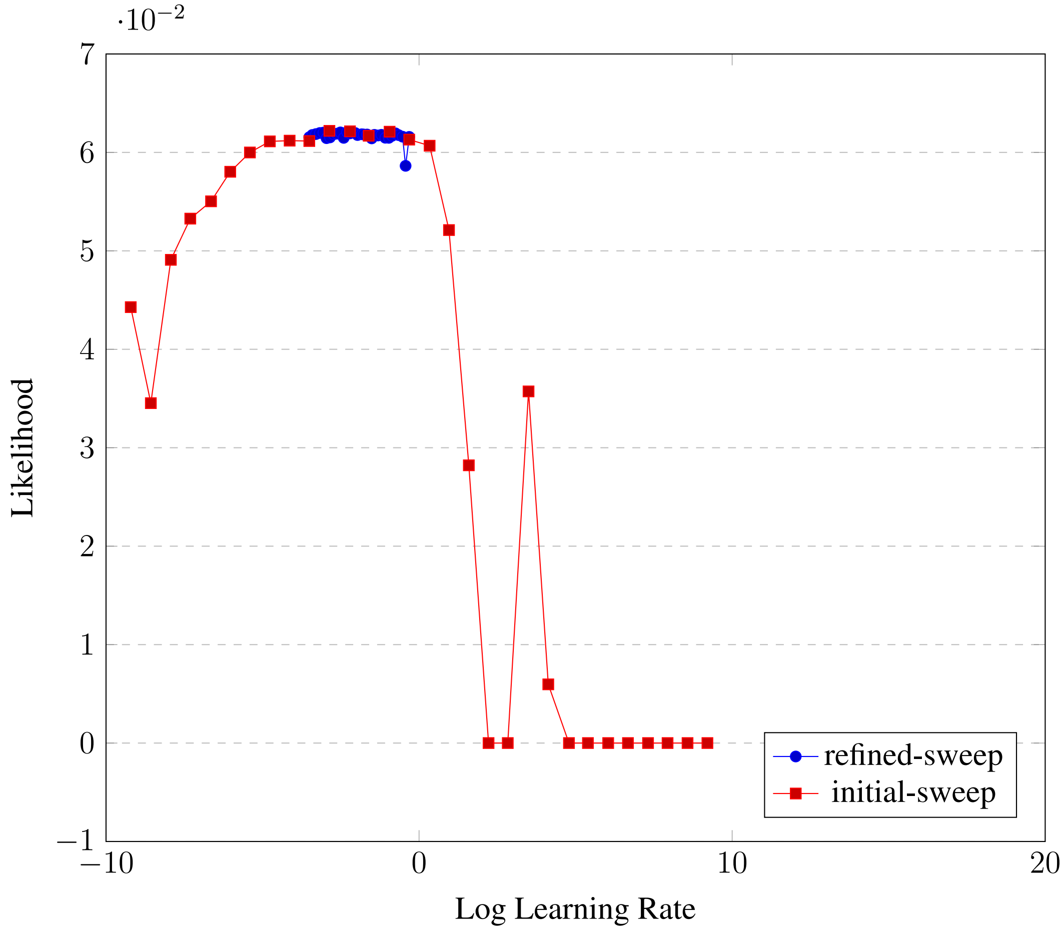


**Figure 4.17.** Example Of Refined Hyper-Parameter Sweep

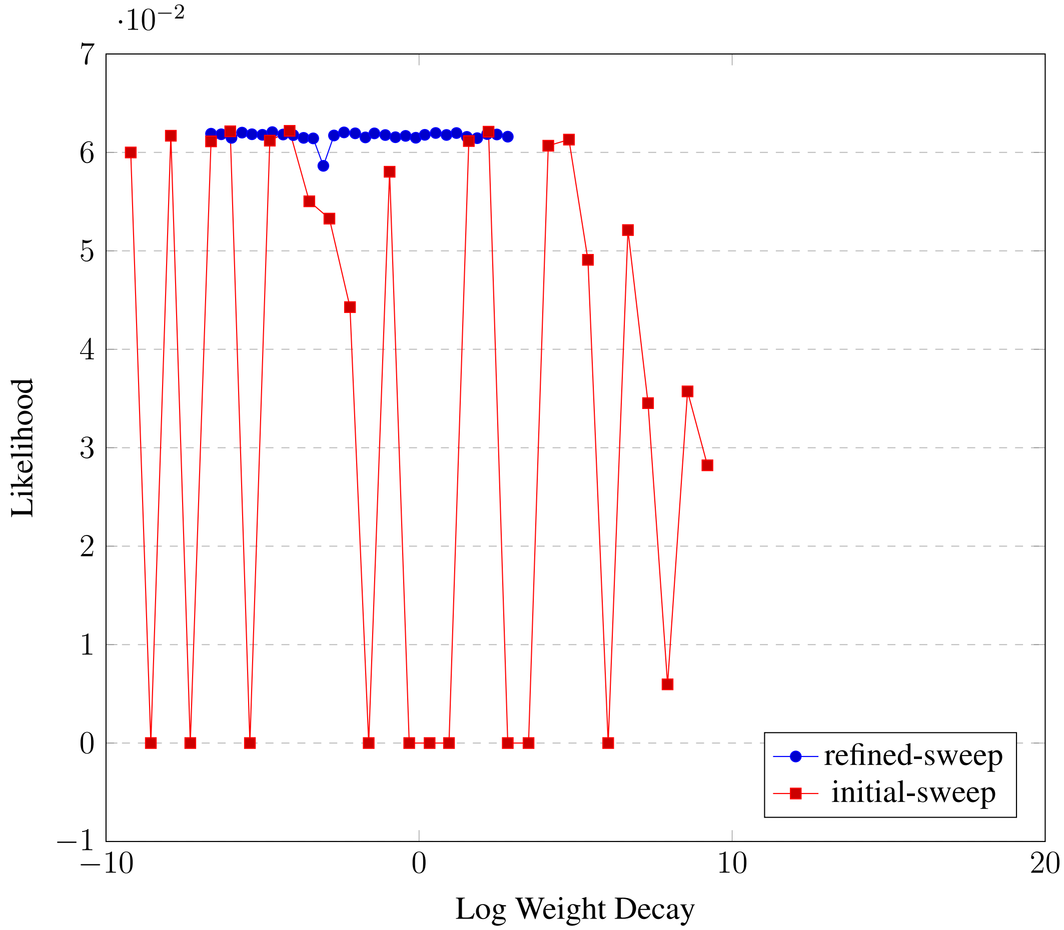


**Figure 4.18.** Example Of Refined Hyper-Parameter Sweep

# Chapter 5

# Structured Ordinal Models

## 5.1 Introduction

Chapters 3 and 4 leveraged a binary, multi-label decomposition of categorical labels into more descriptive binary vectors. These vectors admitted rich, parsimonious models that allowed finer-grained quantitative analysis and richer qualitative introspection than the categorical models that dominated the field at the time of writing. The binarization also helped me avoid dealing with the distinction between low-magnitude ratings and judgments that the questions were *not-applicable* in a particular context.

I also showed how the decompositional labels and the CRF-based probabilistic models allow productive *conditional* inference when information about some subset of the semantic properties is available.

Despite the simplicity of binary models for SPR, it is unsatisfying that they fail to leverage finer-grained information given by annotators—a five-way likert-like rating from "very unlikely" to "very likely" with an additional binary label asking if the question made sense.[1] Since binarization came at the cost of coarsening that signal, I hypothesized that models could make better predictions if trained on this more detailed signal. I expected the finer-grained training data would be important if evaluated against a finer-grained gold standard, but also wondered if the graded signal can even improve prediction with respect to the binarized evaluation?

A straight-forward approach to model all of the data would be to simply replace the binary labels with 6-way categorical labels for each property. Features and modeling could essentially remain otherwise unchanged. However, ignoring the relationships between nearby ordinal labels forfeits potentially helpful inductive bias. I expected that more parsimonious models based on ordinal regression would allow more generalizable models at least in conditions when less data was available.

How can we incorporate a graded signal into the learning process while still maintaining a discrete, joint, probabilistic model of properties conditioned on arbitrary features of the input? The general task of identifying each of a subset of possible properties with grades of inclusion has been called "graded multi-label prediction" (Laghmari, Marsala, and Ramdani, 2016) and can be seen as a multi-label generalization from ordinal regression or as an ordinal generalization from multi-label classification.

Early work in graded multi-label prediction advocated two general reductions to previously known methods: the vertical reduction trains a series of binary multi-label models, one for each level of graded label. The horizontal approach trains a separate ordinal model

[1] The annotation system only gave the option of *not-applicable* when an ordinal rating fell within the bottom three labels.

for each variable. The vertical approach requires a way to ensure that inclusion is monotonically decreasing and enforces an alignment between all labels. Furthermore, the default application of the reduction does not allow dependencies between multi-label classifiers in the vertical reduction nor does it suggest how one would incorporate dependencies between the ordinal classifiers of the horizontal reduction.

In this chapter, I first review some insights and work from the literature related to graded prediction and then give a CRF formulation of popular univariate ordinal models, contribute additional models, and give a simple recipe for composing such univariate ordinal sub-models into larger joint CRF models capable of the conditional queries investigated in Chapter 4. All of the presented models can be represented as a collection of multiplicative factors amenable to approximate inference with the belief propagation algorithm, however, the experiments in this chapter continue with tractable subsets of the properties, allowing me to use brute-force, exact inference to clearly compare the behavior of the various ordinal models. Leveraging automatic differentiation to train the CRFs under approximations allows for additional modeling opportunities within this framework which I leave for future work.

## 5.2 Independent Ordinal Models

Since CRFs offer a generalization of logistic regression to the multivariate scenario, it is natural to turn to univariate ordinal models for inspiration. In this section, I identify some of popular univariate ordinal models and show how they can be represented within the CRF framework.[2] This formulation will also suggest ways within the same framework to jointly

[2] See Section 5.4 for additional background on ordinal approaches.

model multiple ordinal variables. Such a representation is especially appealing since the resulting models are amenable to incorporation with other ordinal or non-ordinal variables and factors and are capable of conditional inference based on partial observations.

The CRF models I consider for a single ordinal variable differ with respect to the following characteristics:

a) The distribution being modeled: e.g. $P(Y = y|X = x)$, $P(Y \geq y|X = x)$, or $P(Y = y, H = h|X = x)$. Here, $X$ represents any variables that are always observed, $Y$ represents the set of output variables for which there is annotation available only at training time, and $H$ represents any hidden (i.e. latent) variables that are modeled but not observed even during training. The output variables $Y$ might include only a single categorical variable $Y$ or might, for example, include multiple binary variables for each possible output $Y_0, Y_1, ...Y_{\text{max}(Y)}$. The important thing is that all such variables must be fully observed for the training examples.

b) The subsets of variables having factors (i.e. the cliques to model).

c) The factors used to model each clique—specifically, the parameterized potential function family for each factor. In practice, I define *log*-potential functions $\phi_i$ to ensure that the corresponding potential function is non-negative. Related to this is the choice of how factor parameters are derived from input features. For the most part, the factors used in my experiments define log-potentials as a linear combination of an input feature vector plus an optional bias; however, with the help of tx, log-potentials can easily be computed through any differentiable neural computation available within pytorch.

Although I exclusively consider log-likelihood training (possibly approximate), tx would also allow investigation of other objectives downstream of inference and automatic differentiation would allow such objectives to be used as a training signal.[3]

## 5.2.1 Nominal Model

The binary approach loses any distinction between ordinal labels that are mapped to the same binary label. Multinomial logistic regression is a common alternative that is nearly as simple mathematically, but requires more parameters. It ignores the ordering of the labels by using a nominal multi-class model and parameters are fit so as to maximize the inferred marginal log probability of the gold *ordinal* label given the input pattern.

The nominal representation is our *least parsimonious* baseline where no effort is made to represent the ordinal variable in any lower dimensional space than the $k-1$ dimensions suggested by a minimal representation. I model $P(Y=y|X=x)$ with a single factor: $\phi(y,\boldsymbol{x}) = \boldsymbol{\theta_y^T} f(\boldsymbol{x}) + b_y$. The "minimal" representation is to fix $\boldsymbol{\theta_0} = \mathbf{0}$ and $b_0 = 0$ since the probability of the first label can be computed deterministically given the probabilities of other labels.

## 5.2.2 Binary

The univariate analog of Chapter 4 is a binary model trained to maximize the inferred marginal log probability of the gold *binary* label given the input pattern. If the gold labels

[3] For example, for ordinal variables corresponding to likelihood judgments, the model could be trained to minimize the squared distance between the inferred conditional probability of a binary variable under the model and a probability parameter (fixed or learned) corresponding to the gold label. Perhaps label 2 of 5 should mean 25 percent probability and the model fits parameters so that inputs labeled as 2 should receive a probability of 25 percent under the model.

are actually ordinal, then a binary model requires thresholding ordinal values to obtain binary labels (as I did in Chapter 4). Given a thresholding scheme on the labeled data, this is simply binary logistic regression.

The binary representation is our *most parsimonious* representation. However, when ordinal observations are available for training, the binary model is limited in that (1) it ignores distinctions between some labels and (2) it requires a hard-coded mapping from observed labels to a training signal; for example, I treat all ordinal labels below some threshold $y_+$ as negative labels and all others as positive[4]:

$$B(y) = \begin{cases} 1, & \text{if } y \geq y_+ \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

**Binary Labels:** In Chapters 3 and 4, we actually replaced the decompositional predictive task with a multi-label binary prediction task. Training and evaluation data was preprocessed to support this paradigm.

**Binary Features:** Alternatively, we can construct a legitimate multi-grade model by adjusting the nominal factor function to only allocate features for distinguishing whether values are above or under a fixed threshold.

My Binary Features model, then, has only a single factor with the following log-potential:

$$\phi(y, \boldsymbol{x}) = \begin{cases} \boldsymbol{\theta^T} f(\boldsymbol{x}) + b, & \text{if } B(y) = 1 \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

[4] In Chapter 4, we used $y_+ = 4$ resulting in only labels of 4 and 5 being considered as positive.

Because we saw that the bias terms are so helpful (especially in prediction that can condition on some observed property labels), a *N*ominal Bias could be added to any binary model (and binary bias could be dropped) in a way that would contribute heavily to the expressiveness of the model while adding very few additional parameters to the model.

**Latent Binary:** A latent binary model introduces an additional binary variable for each ordinal variable in the model. The features of the model are used to predict the value of this latent variable and a bias-only factor between the latent variable and the corresponding ordinal is used to predict the ordinal variable solely using the value of the latent variable as evidence. That is, the latent binary model defines a probability distribution $P(Y = y, H = h|X = x)$ with three separate factors or as a single, combined factor. During inference (and learning) the likelihood of the data is computed by marginalizing over the binary values of $H$.

1)

$$\phi_H(h, \boldsymbol{x}) = \begin{cases} \boldsymbol{\theta^T} f(\boldsymbol{x}) + c, & \text{if } h = 1 \\ \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

2)

$$\phi_{HY}(h, y) = \begin{cases} 0, & \text{if } h = 0 \text{or } y = 0 \\ \\ a_y, & \text{otherwise} \end{cases} \tag{5.4}$$

3)

$$\phi_Y(y) = \begin{cases} 0, & \text{if } y = 0 \\ \\ b_y, & \text{otherwise} \end{cases} \tag{5.5}$$

4) (combined; $a_{y=0} = b_{y=0} = 0$ and $a_{y=\text{max}(Y)} = 1$)

$$\phi_{HY}(h, y, \boldsymbol{x}) = (h - 1)(\boldsymbol{\theta^T} f(\boldsymbol{x}) + c + a_y) + b_y \tag{5.6}$$

If we accept that the ordinal ratings correspond to various levels of probabilities of particular outcomes or assessments, then it is reasonable to represent such an outcome or assessment explicitly as a binary latent variable with predictive features and pairwise factors between each to the corresponding observed ordinal variable (which is not given any predictive features other than a bias term, forcing the model to represent the distribution over ordinal labels solely in terms of feature on the binary variables plus a few parameters to express the mapping from binary to ordinal and an optional prior preference for ordinal labels). The potential function between the observed ordinal and the latent binary factor can either be learned or fixed (as suggested by experiments of Beckham and Pal (2016)). Learning this potential function introduces an additional non-linearity into the model.

This formulation is amenable to inclusion within a larger CRF by simply allowing the latent binary variable to act as a surrogate ordinal variable to the rest of the model. Therefore, pairwise factors between multiple ordinal variables would be represented as pairwise factors between the latent binary variables. Note that there is nothing in the model that enforces the positive values of the latent binary variables correspond to larger ordinal values.

**Linear Latent Binary:** If we choose to fix the pairwise bias parameters $a_y$ between the binary variable and ordinal variable, one option is to use the following linear setting:

$$a_y = \frac{y}{\text{max}(Y)} \tag{5.7}$$

### 5.2.3 Stereotype Model

Anderson's one-dimensional, stereotype model (Anderson, 1984) uses fewer parameters than the nominal model and resembles the latent binary model by supposing that values of the variable lie along a single, latent dimension but avoids the need for a latent variable. Specifically, it models $P(Y = y|X = x)$ with a single factor: $\phi(y, \boldsymbol{x}) = a_y \boldsymbol{\theta}^{\boldsymbol{T}} f(\boldsymbol{x}) + b_y$. The additional parameters $\boldsymbol{a}$ associate each ordinal label with a position along the latent dimension, fixing, according to convention, $a_0 = 0$ and $a_{\text{max}(Y)} = 1$ for identifiability.

Anderson's stereotype model is a log-linear model of ordinal variables where the probability of an ordinal label is given as follows (subject to the constraints that $0 = a_0 < a_1 < ... < a_{\text{max}(Y)} = 1$):

$$P(y|x) \propto \exp\left[a_y \theta^T f(x) + b_y\right]$$

**Linear Stereotype Model:** A specialization of the Stereotype model further constrains $a_y = \frac{y}{\text{max}(Y)}$ for each label $y$.

**Partially-Ordinal Anderson Model:** Sometimes labels are only partially ordinal. That is, there is a set of possible labels to choose from and each input will receive exactly one of these labels and yet not all of the labels are directly comparable to one another. It is possible to extend the stereotype models to handle partially ordinal variables. A partial order can be represented as the transitive reduction of a connected, directed acyclic graph where there is a node for each of the possible values the variable can take. An edge from node $i$ to node $j$ specifies that value $i$ precedes value $j$ in the partial order.

The stereotype model offers a natural extension to accommodate partially ordinal labels and even more elaborate label relationships that fall between ordinal and nominal assumptions. In fact, in the original paper, Anderson (1984) proposed just such a general model using *multiple* latent dimensions. The number of dimensions and the order and position of each label within each latent dimension was of interest in assessing the hypothesis that the data was indeed ordinal as opposed to nominal.

To apply such a strategy to partially-ordinal models, each the factor would have a feature vector for each latent dimensions and each label value would have a (possibly-learned) corresponding value indicating how the magnitude of that label along that latent dimension. One way to choose the number of latent dimensions and to constrain the coefficients given a partial order on a subset of labels would be to compute the transitive reduction of the partial order relationship DAG and assign a latent dimension (i.e. feature id) for each source in the DAG. For any label, then, the set of applicable features would be its set of ancestor source node ids and the corresponding values for those features would be the length of the shortest path from each respective ancestor source. The resulting feature vectors from multiple such graphs could likewise be concatenated for further generalization. Such dependencies between multipliers could be used to inject inductive bias beyond what was done in Anderson's model.

### 5.2.4 Proportional Odds

Cumulative Link Models have been among the most common ordinal regression models and are related to Thurstonian models of ranking (Thurstone, 1927). The generative story in these models is that $k-1$ scalars are chosen as "thresholds" $\theta_1$, ... $\theta_{\max(Y)}$ that partition the real-line into bins corresponding to the $k$ ordinal values. Then, for each input example (represented by features $X_i$), a real-valued scalar $S_i$ is sampled and the corresponding bin is assigned to be its ordinal label.

The probability of an ordinal label given observed input features can be specified in terms of a so-called link model of the scalar given the observed input features: $P(S_i|X_i)$. Specifically, $P(Y_i \geq j|X_i) = P(S_i \geq \theta_j|X_i)$ — that is, the probability that the ordinal label of an example is at least some value $j$ is defined to be the probability that the corresponding latent scalar $S_i$ is at least at large as the corresponding latent scalar threshold.

For example, $S_i$ can be modeled as a logistic random variable with mean given as an affine function of the observed features $f(x_i)$. Work has also investigated training a neural network to learn non-affine functions of observations features (Fernández-Navarro, 2017). Different models of the latent $S_i$ correspond to different, so-called, "link" functions which relate observed features to a probability distribution over $S_i$, therefore, each link function also corresponds to a different ordinal regression method. Christensen (2018) describes several possible link functions and implements the corresponding ordinal regression, supporting extensions for constraining the learned thresholds (e.g. require symmetry), and modeling partial regression effects (i.e. effectively allowing the thresholds to also shift based on observation features) and scale effects (modeling the dispersion of $S_i$ conditioned on observed features as well).

I focus on the cumulative *logit* model. Since the ratio of cumulative odds under this model (i.e. $\frac{P(Y_i \geq k|X_i = \boldsymbol{x})}{1 - P(Y_i \geq k|X_i = \boldsymbol{x})}$) between some $\boldsymbol{x}$ and some other $\boldsymbol{x}'$ is the same for all $k$, McCullagh (1980) calls it the *proportional odds* model. That is, the logarithm of the ratio between the odds of $Y \geq j|X = x_1$ and the odds of $Y \geq j|X = x_2$ is exactly *proportional* to the difference between the two sets of features (i.e. $\boldsymbol{x} - \boldsymbol{x}'$).

Like the Stereotype model, the proportional odds model uses one shared latent dimension and has label-specific bias terms. However, with proportional odds, we model the probability $P(Y \geq y|X = x)$ rather than $P(Y = y|X = x)$. Such a model can be viewed as a CRF with a separate binary variable $D_j$ for $j \in \{1, 3, ..., \max(Y)\}$ such that $D_j = 1 \Leftrightarrow Y \geq j$ and $D_j = 0 \Leftrightarrow Y_i < j$.

In order to include the original graded variable $Y$ in the model, we can simply add a hard factor between the graded variable and each of the label variables which assigns uniform potential to all configurations that abide this relationship and a 0 potential ($-\infty$ log potential) for any that violate it.

Each binary variable receives its own bias term, but all labels share input features and learned weights:

$$\phi_j(d_j, \boldsymbol{x}) = \begin{cases} \boldsymbol{\theta^T} f(\boldsymbol{x}) + b_j, & \text{if } d_j = 1 \\ 0, & \text{otherwise} \end{cases} \tag{5.8}$$

As long as the learned bias terms $\boldsymbol{b}$ are monotonically decreasing, the binary predictions of $H_j$'s will likewise decrease monotonically as required. Cao, Mirjalili, and Raschka (2019) show that maximum likelihood estimates will naturally lead to monotonic $\boldsymbol{b}$s without imposing explicit constraints. They further prove generalization bounds that show that good performance with respect to the binary classifiers translates to good performance with respect to a cost matrix formulation. $\mathcal{C}$.[5] They also suggest a variation of Li and H.-t. Lin (2007) and H.-T. Lin (2008) which removes the constraints on the cost matrix and, in fact, does not use the cost matrix during optimization at all (although they do allow for

[5] Interestingly, although the cost matrix $\mathcal{C}$ appears in the generalization bound to show that their the bound is satisfied for any $\mathcal{C}$, the cost matrix does not appear as a term in the optimization algorithm.

class-specific weights). Interestingly, in the case of uniform class weights, their model is exactly the proportional odds model where $k-1$ binary variables are modeled as follows:[6]

$$P(Y \geq j | X = x) = \frac{1}{1 + \exp\left[-(a_j + X\beta)\right]}$$

They use a convolutional neural network (CNN) to form their feature representation, and although they cite the paper of McCullagh (1980), but it was not clear to me that they were making an explicit connection between the two models. In fact, the proposed coding for ordinal variables goes back as far as Walter, Feinstein, and Wells (1987) who propose the same for *independent* variables, while McCullagh (1980) introduces the representation as a way of describing models of ordinal *dependent* variables.

**Collapsed Proportional Odds:** Based on the features available to the Proportional Odds model, I consider a model that applies those features directly to the graded variables rather than introducing additional variables, so I think of it as a "collapsed" version of the Proportional Odds model. It is equivalent to the Binary Features model under a threshold of $1$ and using a nominal bias rather than a binary bias. It is equivalent to a stereotype model where the multipliers are fixed to be 0 for any configuration including a 0 label and 1 otherwise.

Figures 5.1 and 5.2 summarize the CRF representation of unary ordinal models discussed here.

[6] As suggested in Harrell (2015), I describe the probabilities in terms of $\geq$ so that the large $a$s correspond to larger bias toward higher ordinal values of the variable

## 5.3 Joint Models

Having reviewed a number of traditional and custom models for unary probabilistic models of ordinal regression, I now demonstrate how to create higher-order analogs.

**Nominal, Binary Labels, Binary Features, and their minimal variants:** Features and bias (including the optional nominal bias) are available for joint configurations in the same way that they were available for unary configurations in the univariate model. For minimal representations, lower-order subsets are modeled first and the log-potential is fixed to 0 for any configuration containing *any* 0 label.

**Latent Binary:** Higher-order factors connect latent binary variables rather than graded variables. If applicable, nominal bias is applied to higher-order graded variable groups.

**Stereotype and Linear Stereotype:** A separate feature-based score is computed for each subset of variables which corresponds to the number of binary configurations available for that number of variables. My implementation follows the minimal representation where smaller cliques are assumed to have already been modeled meaning that only a single feature vector need be added for any clique in order to model the joint affinity of all of the variables. A nominal bias and either a fixed or learned nominal multiplier is also given (again, I use the minimal representation where any multiplier or bias involving any 0-label is fixed to a log-potential of 0 and the multiplier corresponding to the last configuration (i.e. $Y = (y_{\max(Y_0)}, y_{\max(Y_1)}, ...y_{\max(Y_n)}))$ is fixed to 1.

**Proportional Odds:** Label variables are created for unary factors and reused in other factors. A joint graded minimal configuration (i.e. one not including any zero labels) corresponds to a selection of binary label variable from each participating graded variable in the factor. The weight for all such configuration groups are shared in the proportional odds model so it is generated once for a given graded clique and reused for all configurations of the clique. An additional bias only factor is also generated for each configuration.

**Collapsed Proportional Odds:** This can be handled either as the binary features or stereo type higher-order extension (both are equivalent).

The general principle is to ensure that higher-order configurations have the same awareness or invariance to label distinctions as did the univariate sub models.

In this section I compare the ability of the various ordinal models for the subsets of the SPRL task. As in Chapter 4, I use a brute-force inferencer for exact inference and focus on only subsets of the SPR properties.

First I evaluate the various models on their ability to model the *volition* SPR property. I follow the same hyper-parameter search process described in Chapter 4 using the same datasets except that the batch-size was used as a hyper-parameter for these models since some result in non-convex objective functions, so mini-batches allows additional opportunities to escape from local optima. I consider the range (in log-space) from 100 to 10000 sized batches, rounding to a single significant digit).

**Table 5.1.** Clique model comparison (averaged over two runs of the experiment) (evaluated on val data) for a size-1 subset of the properties.

Each row represents the average over training data sizes [1000, 2000, 3000, 4000, 5000, 6338]. Models with $^{-}$ only include bias terms. Models with $^{+}$ include a bias on all ordinal interactions rather than binary interactions. Models with $^{*}$ have redundant parameters (i.e. non-minimal). Binary-Label models are trained and evaluated against the threshold labels. 'L' columns give the exponentiated average log-likelihood of a joint labeling (either binary or nominal) under the model.

| Model | Param Count | $\text{L}_{\text{BIN}}$ | L | $\text{F1}_{\text{BIN}}$ | $\text{K}_{\text{NOM}}$ | $\text{K}_{\text{ORD}}$ | $\text{K}_{\text{INT}}$ | $\text{K}_{\text{RAT}}$ | $\rho_{\text{P}}$ | $\rho_{\text{S}}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $\text{Binary}^{-}_{\text{LAB}}$ | 1 | 52.4 | - | 0.0 | - | - | - | - | - | - |
| $\text{Nominal}^{-}$ | 4 | - | 43.4 | 0.0 | -20.9 | -23.6 | -22.8 | -23.3 | - | - |
| $\text{Binary}_{\text{FEAT}}$ | 4097 | - | 29.9 | 85.5 | 22.0 | 72.3 | 74.7 | 76.2 | 79.0 | 79.1 |
| $\text{Binary}_{\text{LAB}}$ | 4097 | **77.8** | - | 85.7 | - | - | - | - | - | - |
| $\text{Binary}^{\text{LIN}}_{\text{LAT}}$ | 4097 | - | 22.2 | 56.0 | -21.8 | -23.6 | -23.9 | -22.5 | 20.4 | 21.6 |
| $\text{Binary}^{+}_{\text{FEAT}}$ | 4100 | - | 64.5 | 85.8 | 74.6 | 79.5 | 79.5 | 77.3 | 79.5 | 79.5 |
| $\text{Binary}^{\text{LIN+}}_{\text{LAT}}$ | 4100 | - | 50.7 | 55.8 | 40.6 | 42.6 | 43.2 | 40.9 | 76.9 | 76.7 |
| PropOdds | 4100 | - | 64.4 | 85.1 | 74.2 | 79.2 | 78.9 | 77.4 | 78.9 | 79.3 |
| $\text{PropOdds}_{\text{C}}$ | 4100 | - | 66.2 | 85.6 | 75.3 | **80.6** | 80.1 | **79.0** | 80.1 | **80.6** |
| $\text{Stereo}^{\text{LIN}}$ | 4100 | - | 67.0 | 85.9 | 75.1 | 80.2 | 80.1 | 78.3 | 80.1 | 80.3 |
| $\text{Binary}_{\text{LAT}}$ | 4101 | - | 44.1 | 0.0 | -20.9 | -23.6 | -22.8 | -23.3 | - | - |
| Stereo | 4103 | - | 67.4 | 86.0 | **75.3** | 80.5 | **80.2** | 78.6 | **80.3** | 80.5 |
| $\text{Binary}^{+}_{\text{LAT}}$ | 4104 | - | 65.4 | 85.9 | 74.9 | 79.9 | 79.8 | 78.0 | 79.8 | 80.0 |
| $\text{Binary}^{*}_{\text{LAB}}$ | 8194 | 77.8 | - | 85.6 | - | - | - | - | - | - |
| Nominal | 16388 | - | 67.1 | **86.1** | 74.7 | 80.0 | 80.1 | 77.7 | 80.1 | 80.0 |
| $\text{Nominal}^{*}$ | 20485 | - | **67.9** | 86.0 | 74.9 | 80.1 | 80.1 | 78.0 | 80.1 | 80.2 |

**Table 5.2.** Clique model comparison (averaged over two runs of the experiment) (evaluated on dev data) for a size-1 subset of the properties

Each row represents the average over training data sizes [1000, 2000, 3000, 4000, 5000, 6338]. Models with $^{-}$ only include bias terms. Models with $^{+}$ include a bias on all ordinal interactions rather than binary interactions. Models with $^{*}$ have redundant parameters (i.e. non-minimal). Binary-Label models are trained and evaluated against the threshold labels. 'L' columns give the exponentiated average log-likelihood of a joint labeling (either binary or nominal) under the model.

| Model | Param Count | $L_{\text{BIN}}$ | L | $F1_{\text{BIN}}$ | $K_{\text{NOM}}$ | $K_{\text{ORD}}$ | $K_{\text{INT}}$ | $K_{\text{RAT}}$ | $\rho_P$ | $\rho_S$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Binary$^{-}_{\text{LAB}}$ | 1 | 53.1 | - | 0.0 | - | - | - | - | - | - |
| Nominal$^{-}$ | 4 | - | 41.7 | 0.0 | -19.1 | -22.6 | -21.6 | -22.1 | - | - |
| Binary$_{\text{FEAT}}$ | 4097 | - | 30.4 | 86.5 | 24.7 | 76.4 | 77.9 | 78.5 | 81.9 | 82.1 |
| Binary$_{\text{LAB}}$ | 4097 | 79.7 | - | 86.9 | - | - | - | - | - | - |
| Binary$^{\text{LIN}}_{\text{LAT}}$ | 4097 | - | 22.1 | 53.7 | -22.8 | -25.4 | -25.6 | -23.3 | 20.5 | 22.0 |
| Binary$^{+}_{\text{FEAT}}$ | 4100 | - | 62.8 | **87.0** | **75.7** | 82.6 | 82.5 | 79.5 | 82.6 | 82.6 |
| Binary$^{\text{LIN+}}_{\text{LAT}}$ | 4100 | - | 48.7 | 57.0 | 42.3 | 45.4 | 46.2 | 43.1 | 80.6 | 80.3 |
| PropOdds | 4100 | - | 64.6 | 86.3 | 74.6 | 81.6 | 81.6 | 79.1 | 81.6 | 81.7 |
| PropOdds$_{\text{C}}$ | 4100 | - | 64.1 | 86.3 | 75.1 | 82.3 | 82.0 | **79.9** | 82.1 | 82.3 |
| Stereo$^{\text{LIN}}$ | 4100 | - | 65.9 | 86.4 | 74.9 | 82.0 | 81.9 | 79.2 | 81.9 | 82.0 |
| Binary$_{\text{LAT}}$ | 4101 | - | 43.1 | 0.0 | -19.1 | -22.6 | -21.6 | -22.1 | - | - |
| Stereo | 4103 | - | 66.1 | 86.9 | 75.5 | 82.6 | 82.5 | 79.9 | 82.6 | 82.6 |
| Binary$^{+}_{\text{LAT}}$ | 4104 | - | 63.3 | 86.9 | 75.6 | **82.7** | **82.6** | 79.9 | **82.7** | **82.8** |
| Binary$^{*}_{\text{LAB}}$ | 8194 | **79.7** | - | 86.9 | - | - | - | - | - | - |
| Nominal | 16388 | - | 65.4 | 86.8 | 75.3 | 82.4 | 82.4 | 79.3 | 82.4 | 82.4 |
| Nominal$^{*}$ | 20485 | - | **66.4** | 86.8 | 75.2 | 82.2 | 82.3 | 79.0 | 82.3 | 82.2 |

**Table 5.3.** Clique model comparison – tiny training sizes only (averaged over two runs of the experiment) (evaluated on val data) for a size-1 subset of the properties.
Each row represents the average over training data sizes [10, 20, 30, 40]. Models with $^{-}$ only include bias terms. Models with $^{+}$ include a bias on all ordinal interactions rather than binary interactions. Models with $^{*}$ have redundant parameters (i.e. non-minimal). Binary-Label models are trained and evaluated against the threshold labels. 'L' columns give the exponentiated average log-likelihood of a joint labeling (either binary or nominal) under the model.

| Model | Param Count | $\mathrm{L}_{\mathrm{BIN}}$ | L | $\mathrm{F1}_{\mathrm{BIN}}$ | $\mathrm{K}_{\mathrm{NOM}}$ | $\mathrm{K}_{\mathrm{ORD}}$ | $\mathrm{K}_{\mathrm{INT}}$ | $\mathrm{K}_{\mathrm{RAT}}$ | $\rho_{\mathrm{P}}$ | $\rho_{\mathrm{S}}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $\mathrm{Binary}^{\mathrm{LIN+}}_{\mathrm{LAT}}$ | 4100 | - | 61.1 | 75.0 | 70.3 | 70.3 | 70.3 | 70.3 | **100.0** | **100.0** |
| $\mathrm{PropOdds}_{\mathrm{C}}$ | 4100 | - | **95.4** | **98.2** | **97.4** | **97.4** | **97.4** | **97.4** | 97.5 | 97.5 |
| $\mathrm{Stereo}^{\mathrm{LIN}}$ | 4100 | - | 85.9 | 93.3 | 89.8 | 89.8 | 89.8 | 89.8 | 90.3 | 90.3 |
| $\mathrm{Nominal}^{*}$ | 20485 | - | 90.6 | 90.3 | 86.3 | 86.3 | 86.3 | 86.3 | 88.0 | 88.0 |

**Table 5.4.** Clique model comparison – tiny training sizes only (averaged over two runs of the experiment) (evaluated on dev data) for a size-1 subset of the properties.
Each row represents the average over training data sizes [10, 20, 30, 40]. Models with $^{-}$ only include bias terms. Models with $^{+}$ include a bias on all ordinal interactions rather than binary interactions. Models with $^{*}$ have redundant parameters (i.e. non-minimal). Binary-Label models are trained and evaluated against the threshold labels. 'L' columns give the exponentiated average log-likelihood of a joint labeling (either binary or nominal) under the model.

| Model | Param Count | $\mathrm{L}_{\mathrm{BIN}}$ | L | $\mathrm{F1}_{\mathrm{BIN}}$ | $\mathrm{K}_{\mathrm{NOM}}$ | $\mathrm{K}_{\mathrm{ORD}}$ | $\mathrm{K}_{\mathrm{INT}}$ | $\mathrm{K}_{\mathrm{RAT}}$ | $\rho_{\mathrm{P}}$ | $\rho_{\mathrm{S}}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $\mathrm{Binary}^{\mathrm{LIN+}}_{\mathrm{LAT}}$ | 4100 | - | **41.1** | 54.9 | 37.6 | 40.6 | 41.2 | 39.0 | 63.3 | 63.2 |
| $\mathrm{PropOdds}_{\mathrm{C}}$ | 4100 | - | 7.4 | **74.4** | **58.1** | **63.5** | **63.5** | **61.6** | **64.3** | **64.4** |
| $\mathrm{Stereo}^{\mathrm{LIN}}$ | 4100 | - | 18.6 | 73.7 | 56.7 | 62.7 | 62.8 | 60.5 | 63.5 | 63.4 |
| $\mathrm{Nominal}^{*}$ | 20485 | - | 7.7 | 71.9 | 55.2 | 60.3 | 60.7 | 58.1 | 61.9 | 61.8 |

**Table 5.5.** Clique model comparison (evaluated on val data) for a size-3 subset of the properties. Each row represents the average over training data sizes [1000, 2000, 3000, 4000, 5000, 6338]. Models with $^{-}$ only include bias terms. Models with $^{+}$ include a bias on all ordinal interactions rather than binary interactions. Models with $^{*}$ have redundant parameters (i.e. non-minimal). Binary-Label models are trained and evaluated against the threshold labels. 'L' columns give the exponentiated average log-likelihood of a joint labeling (either binary or nominal) under the model.

| Model | Param Count | $\text{L}_{\text{BIN}}$ | L | $\text{F1}_{\text{BIN}}$ | $\text{K}_{\text{NOM}}$ | $\text{K}_{\text{ORD}}$ | $\text{K}_{\text{INT}}$ | $\text{K}_{\text{RAT}}$ | $\rho_{\text{P}}$ | $\rho_{\text{S}}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Binary$^{-}_{\text{LAB}}$ | 6 | 31.9 | - | 0.0 | - | - | - | - | - | - |
| Nominal$^{-}$ | 60 | - | 21.1 | 0.0 | -22.3 | -25.0 | -24.4 | -24.8 | - | - |
| Binary$_{\text{FEAT}}$ | 24582 | - | 3.3 | 85.8 | 23.1 | 70.6 | 73.7 | 76.1 | 78.8 | 78.6 |
| Binary$_{\text{LAB}}$ | 24582 | 57.0 | - | 85.8 | - | - | - | - | - | - |
| Binary$^{\text{LIN}}_{\text{LAT}}$ | 24582 | - | 1.1 | 54.2 | -39.3 | -42.7 | -42.7 | -40.6 | 7.5 | 8.7 |
| Binary$_{\text{LAT}}$ | 24594 | - | 8.3 | 0.0 | -22.3 | -25.0 | -24.4 | -24.8 | - | - |
| Binary$^{+}_{\text{FEAT}}$ | 24636 | - | 35.6 | 85.4 | 73.5 | 78.3 | 78.4 | 76.3 | 78.6 | 78.5 |
| Binary$^{\text{LIN+}}_{\text{LAT}}$ | 24636 | - | 30.7 | **86.1** | **74.8** | **79.8** | **79.8** | **78.0** | **79.9** | **79.9** |
| PropOdds$_{\text{C}}$ | 24636 | - | 38.4 | 85.5 | 74.2 | 79.2 | 79.1 | 77.8 | 79.2 | 79.3 |
| Stereo$^{\text{LIN}}$ | 24636 | - | 38.9 | 85.9 | 74.4 | 79.4 | 79.4 | 77.8 | 79.5 | 79.5 |
| Binary$^{+}_{\text{LAT}}$ | 24648 | - | 37.7 | 85.8 | 74.2 | 79.1 | 79.1 | 77.4 | 79.2 | 79.2 |
| Stereo | 24690 | - | 39.3 | 85.8 | 74.4 | 79.4 | 79.5 | 77.7 | 79.6 | 79.5 |
| Binary$^{*}_{\text{LAB}}$ | 73746 | **57.2** | - | 85.5 | - | - | - | - | - | - |
| Nominal | 245820 | - | 39.7 | 85.9 | 74.3 | 79.3 | 79.5 | 77.3 | 79.6 | 79.5 |
| Nominal$^{*}$ | 368730 | - | **40.8** | 85.7 | 74.0 | 79.1 | 79.2 | 77.4 | 79.2 | 79.2 |

**Table 5.6.** Clique model comparison (evaluated on dev data) for a size-3 subset of the properties. Each row represents the average over training data sizes [1000, 2000, 3000, 4000, 5000, 6338]. Models with $^{-}$ only include bias terms. Models with $^{+}$ include a bias on all ordinal interactions rather than binary interactions. Models with $^{*}$ have redundant parameters (i.e. non-minimal). Binary-Label models are trained and evaluated against the threshold labels. 'L' columns give the exponentiated average log-likelihood of a joint labeling (either binary or nominal) under the model.

| | Model | Param Count | $\mathrm{L}_{\mathrm{BIN}}$ | L | $\mathrm{F1}_{\mathrm{BIN}}$ | $\mathrm{K}_{\mathrm{NOM}}$ | $\mathrm{K}_{\mathrm{ORD}}$ | $\mathrm{K}_{\mathrm{INT}}$ | $\mathrm{K}_{\mathrm{RAT}}$ | $\rho_{\mathrm{P}}$ | $\rho_{\mathrm{S}}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Binary$^{-}_{\mathrm{LAB}}$ | 6 | 32.8 | - | 0.0 | - | - | - | - | - | - | |
| Nominal$^{-}$ | 60 | - | 20.0 | 0.0 | -20.6 | -23.9 | -23.1 | -23.6 | - | - | |
| Binary$_{\mathrm{FEAT}}$ | 24582 | - | 3.3 | 86.0 | 25.2 | 73.8 | 76.1 | 78.1 | 80.8 | 80.9 | |
| Binary$_{\mathrm{LAB}}$ | 24582 | 57.6 | - | 85.9 | - | - | - | - | - | - | |
| Binary$^{\mathrm{LIN}}_{\mathrm{LAT}}$ | 24582 | - | 1.1 | 52.0 | -40.2 | -44.5 | -44.2 | -41.3 | 8.5 | 9.6 | |
| Binary$_{\mathrm{LAT}}$ | 24594 | - | 7.9 | 0.0 | -20.7 | -23.9 | -23.1 | -23.6 | -1.4 | -1.4 | |
| Binary$^{+}_{\mathrm{FEAT}}$ | 24636 | - | 33.1 | 85.4 | 73.5 | 79.7 | 79.7 | 77.2 | 79.8 | 79.8 | |
| Binary$^{\mathrm{LIN+}}_{\mathrm{LAT}}$ | 24636 | - | 29.3 | 85.9 | 74.6 | 81.0 | 80.9 | 78.7 | 81.0 | 81.2 | |
| PropOdds$_{\mathrm{C}}$ | 24636 | - | 37.8 | 85.2 | 73.9 | 80.5 | 80.2 | 78.9 | 80.3 | 80.5 | |
| Stereo$^{\mathrm{LIN}}$ | 24636 | - | 38.0 | 85.5 | 74.1 | 80.4 | 80.4 | 78.4 | 80.4 | 80.6 | |
| Binary$^{+}_{\mathrm{LAT}}$ | 24648 | - | 36.3 | 85.6 | 74.1 | 80.5 | 80.4 | 78.7 | 80.4 | 80.6 | |
| Stereo | 24690 | - | 37.7 | 85.8 | 74.6 | 81.1 | 81.0 | 79.0 | 81.0 | 81.2 | |
| Binary$^{*}_{\mathrm{LAB}}$ | 73746 | **58.0** | - | 86.0 | - | - | - | - | - | - | |
| Nominal | 245820 | - | 38.0 | **86.0** | 74.6 | 81.0 | 81.0 | 78.6 | 81.1 | 81.1 | |
| Nominal$^{*}$ | 368730 | - | **40.3** | 86.0 | **74.7** | **81.2** | **81.1** | **79.0** | **81.2** | **81.3** | |

**Nominal**

$$P(Y = y|X = \boldsymbol{x}) \quad \propto \exp\left[\theta_y^T f(\boldsymbol{x}) + b_y\right]$$

**Binary Labels**

$$P(H = h|X = \boldsymbol{x}) \quad \propto \exp\left[\theta_h^T f(\boldsymbol{x}) + b_h\right]$$

$$P(Y \geq y_+|X = \boldsymbol{x}) \quad = P(H = 1|X = \boldsymbol{x})$$

**Binary Features**

$$P(Y = y|X = \boldsymbol{x}) \quad \propto \exp\left[\theta_{y \geq y_+}^T f(\boldsymbol{x}) + b_{y \geq y_+}\right]$$

**Latent Binary**

$$P(Y = y, H = h|X = \boldsymbol{x}) \quad \propto \exp\left[\theta_h^T f(\boldsymbol{x}) + c_h + ha_y + b_y\right]$$

$$P(Y = y|X = \boldsymbol{x}) \quad = P(Y = y, H = 1|X = \boldsymbol{x}) + P(Y = y, H = 0|X = \boldsymbol{x})$$

**Linear Regression**

$$y|X = \boldsymbol{x} \quad = \theta^T f(\boldsymbol{x}) + b_y$$

**Stereotype**

$$P(Y = y|X = \boldsymbol{x}) \quad \propto \exp\left[a_y \theta^T f(\boldsymbol{x}) + b_y\right]$$

**Proportional Odds**

$$P(D_j = d_j|X = \boldsymbol{x}) \quad \propto \exp\left[\theta^T f(\boldsymbol{x}) + b_j\right] \qquad 0 < j \leq y_{\text{MAX}}$$

$$P(Y \geq y|X = \boldsymbol{x}) \quad = P(H_y = 1|X = \boldsymbol{x})$$

$$P(Y = y|X = \boldsymbol{x}) \quad = P(Y \geq y|X = \boldsymbol{x}) - P(Y \geq y + 1|X = \boldsymbol{x})$$

**Collapsed Proportional Odds**

$$P(Y = y|X = \boldsymbol{x}) \quad \propto \exp\left[\theta_{y \geq 1}^T f(\boldsymbol{x}) + b_y\right]$$

***Domains***

$$y \in \{0, 1, ..., y_{\text{MAX}}\} \qquad h \in \{0, 1\}$$

***Minimal Variants***

$$a_0 = b_0 = c_0 = 0 \ a_{y_{\text{MAX}}} = 1 \qquad \boldsymbol{\theta_0^T} = \mathbf{0}$$

***Linear Variants***

$$a_y = \frac{a}{\max(Y)}$$

**Figure 5.1.** Unary ordinal model representations

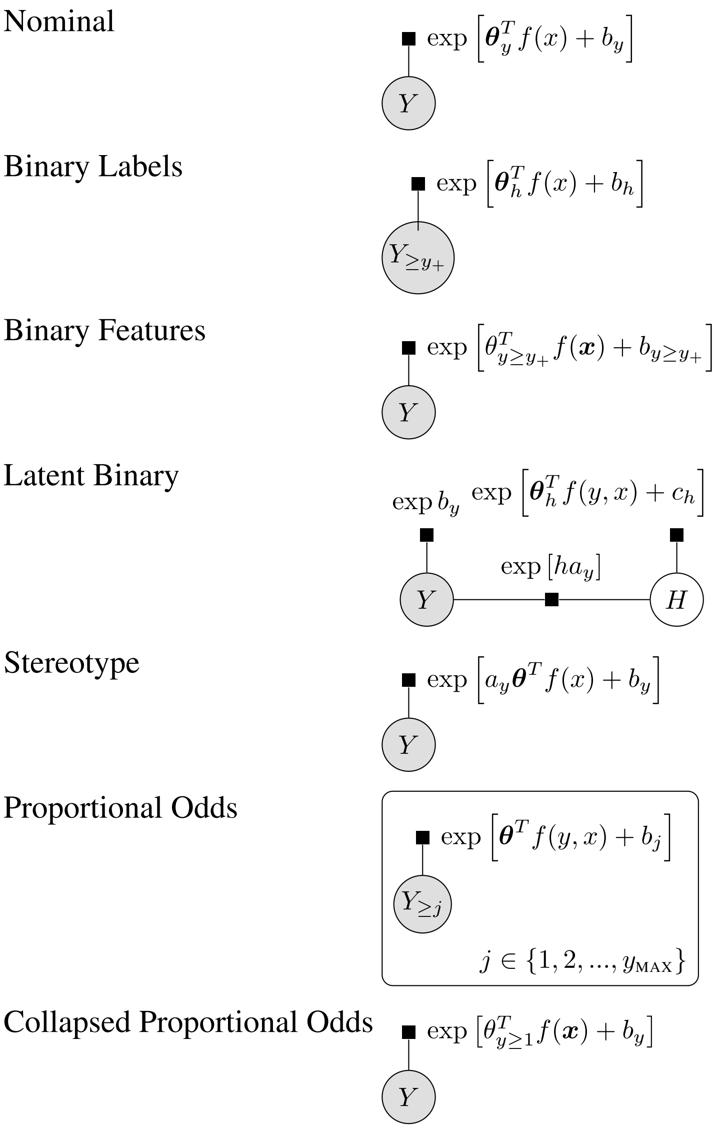

Nominal
$\exp\left[\boldsymbol{\theta}_y^T f(x) + b_y\right]$
$Y$
Binary Labels
$\exp\left[\boldsymbol{\theta}_h^T f(x) + b_h\right]$
$Y_{\geq y_+}$
Binary Features
$\exp\left[\theta_{y \geq y_+}^T f(\boldsymbol{x}) + b_{y \geq y_+}\right]$
$Y$
Latent Binary
$\exp b_y$
$\exp\left[\boldsymbol{\theta}_h^T f(y, x) + c_h\right]$
$\exp\left[h a_y\right]$
$Y$
$H$
Stereotype
$\exp\left[a_y \boldsymbol{\theta}^T f(x) + b_y\right]$
$Y$
Proportional Odds
$\exp\left[\boldsymbol{\theta}^T f(y, x) + b_j\right]$
$Y_{\geq j}$
$j \in \{1, 2, ..., y_{\text{MAX}}\}$
Collapsed Proportional Odds
$\exp\left[\theta_{y \geq 1}^T f(\boldsymbol{x}) + b_y\right]$
$Y$

**Figure 5.2.** Ordinal Graphical Modals

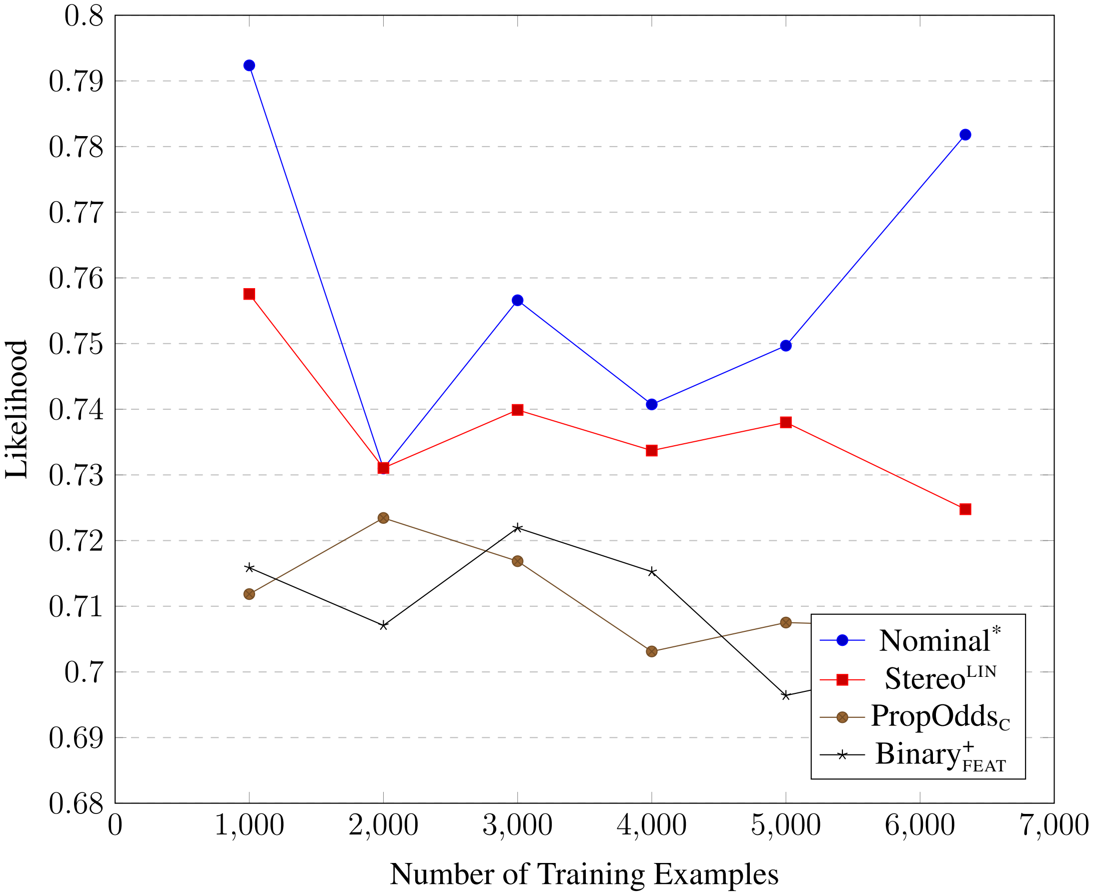


**Figure 5.3.** Clique model comparison of likelihood (averaged over two runs of the experiment) (evaluated on train data) for a size-1 subset of the properties.

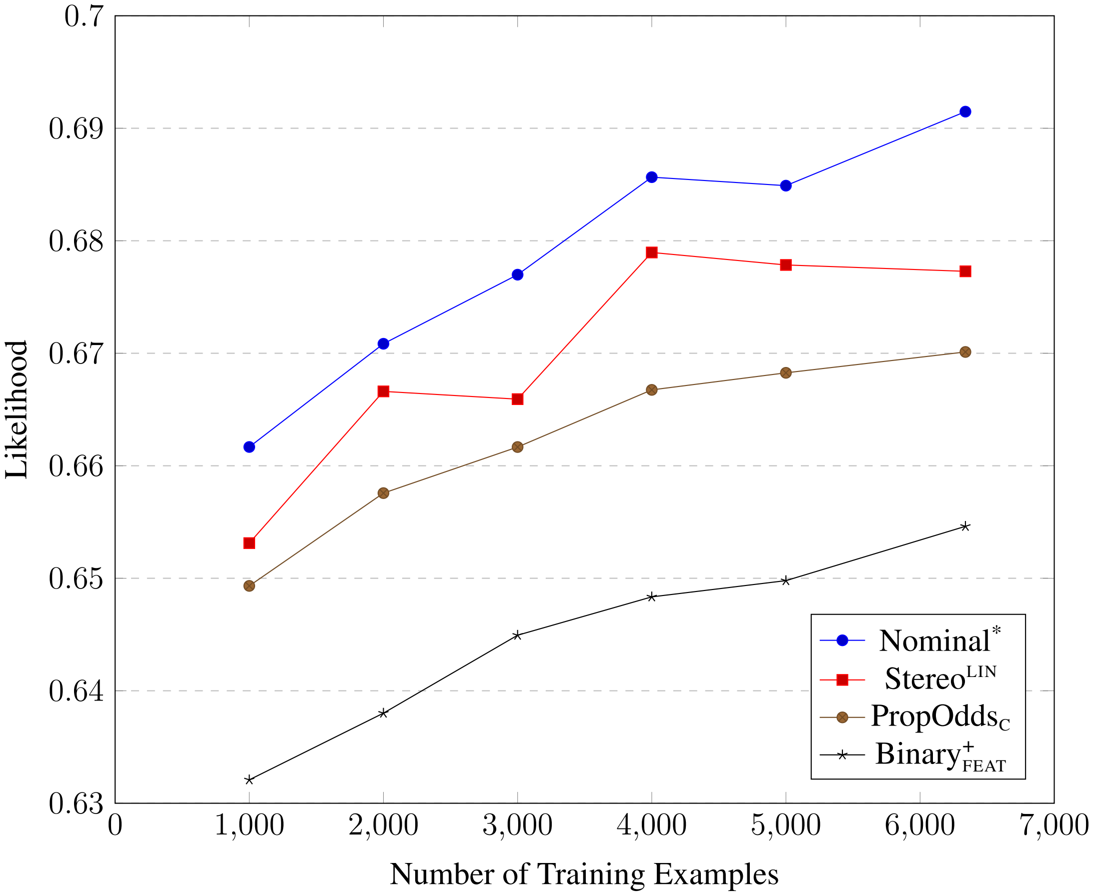


**Figure 5.4.** Clique model comparison of likelihood (averaged over two runs of the experiment) (evaluated on val data) for a size-1 subset of the properties.

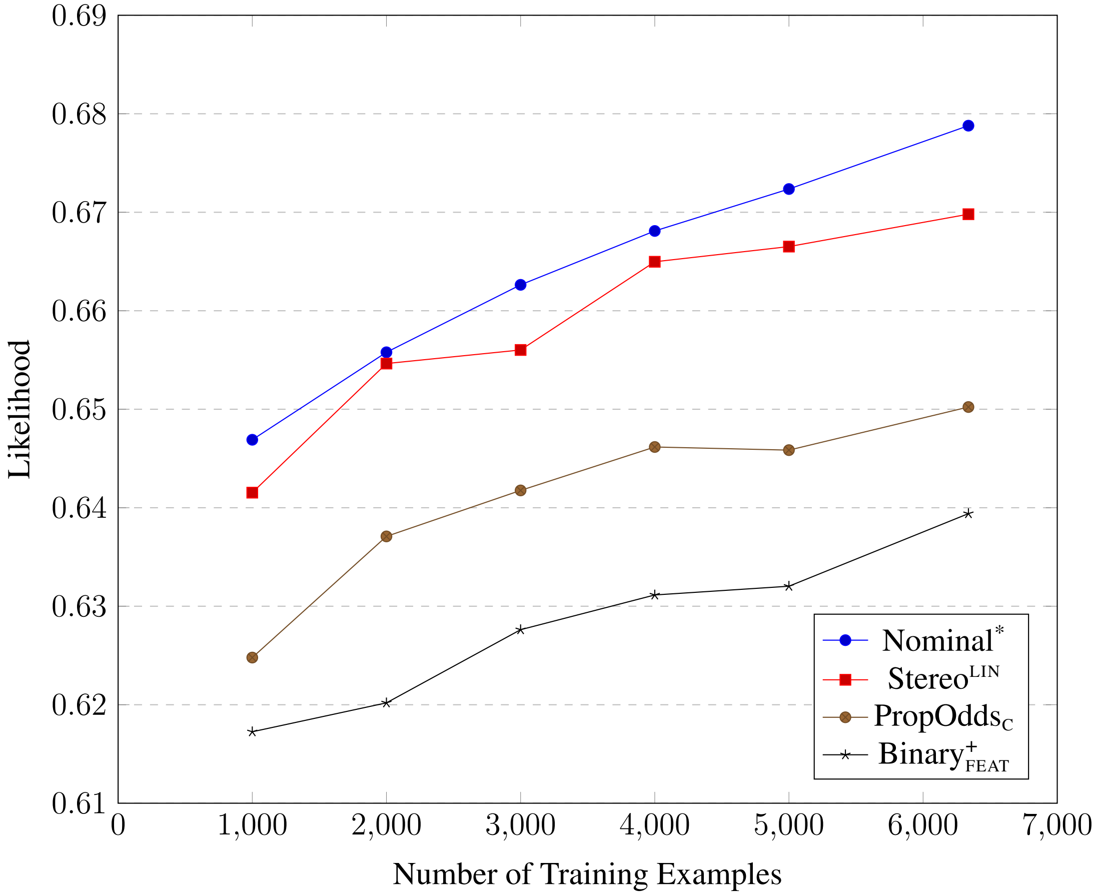


**Figure 5.5.** Clique model comparison of likelihood (averaged over two runs of the experiment) (evaluated on dev data) for a size-1 subset of the properties.

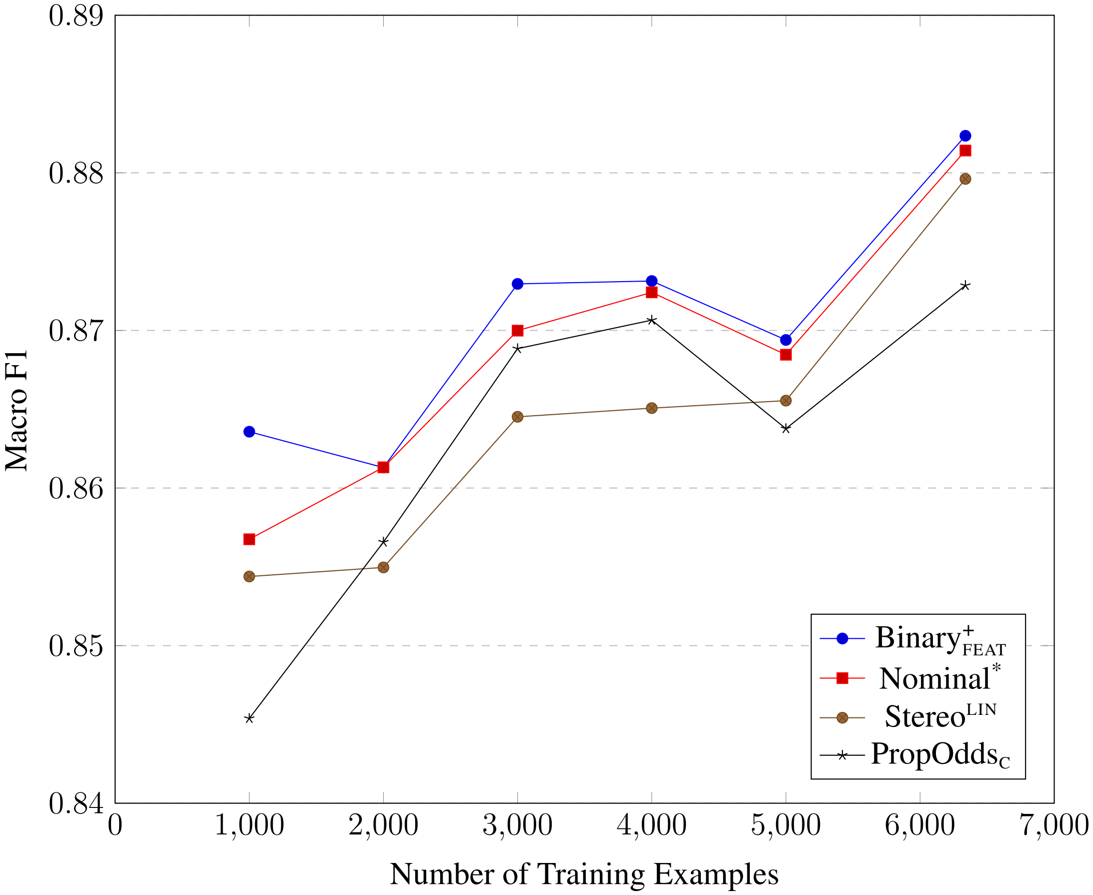


**Figure 5.6.** Clique model comparison of macro-f1 (averaged over two runs of the experiment) (evaluated on dev data) for a size-1 subset of the properties.

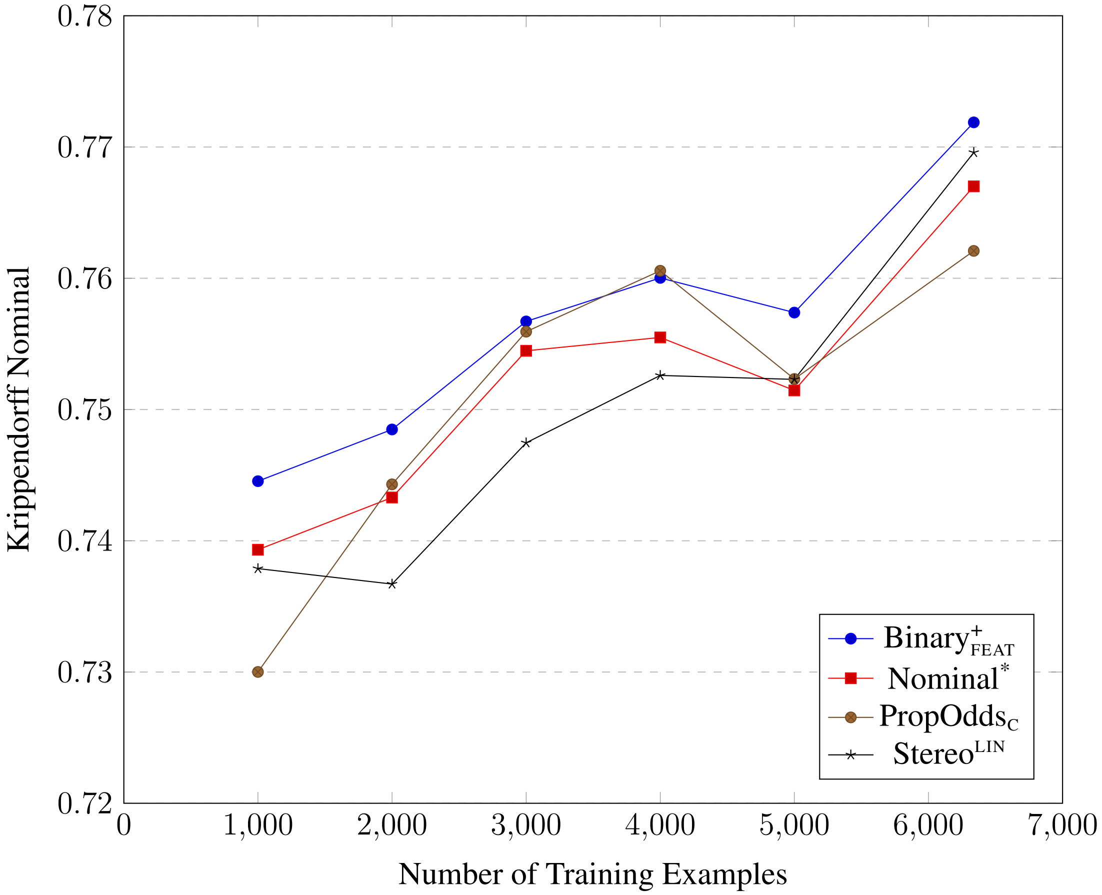


**Figure 5.7.** Clique model comparison of krippendorff-nominal (averaged over two runs of the experiment) (evaluated on dev data) for a size-1 subset of the properties.

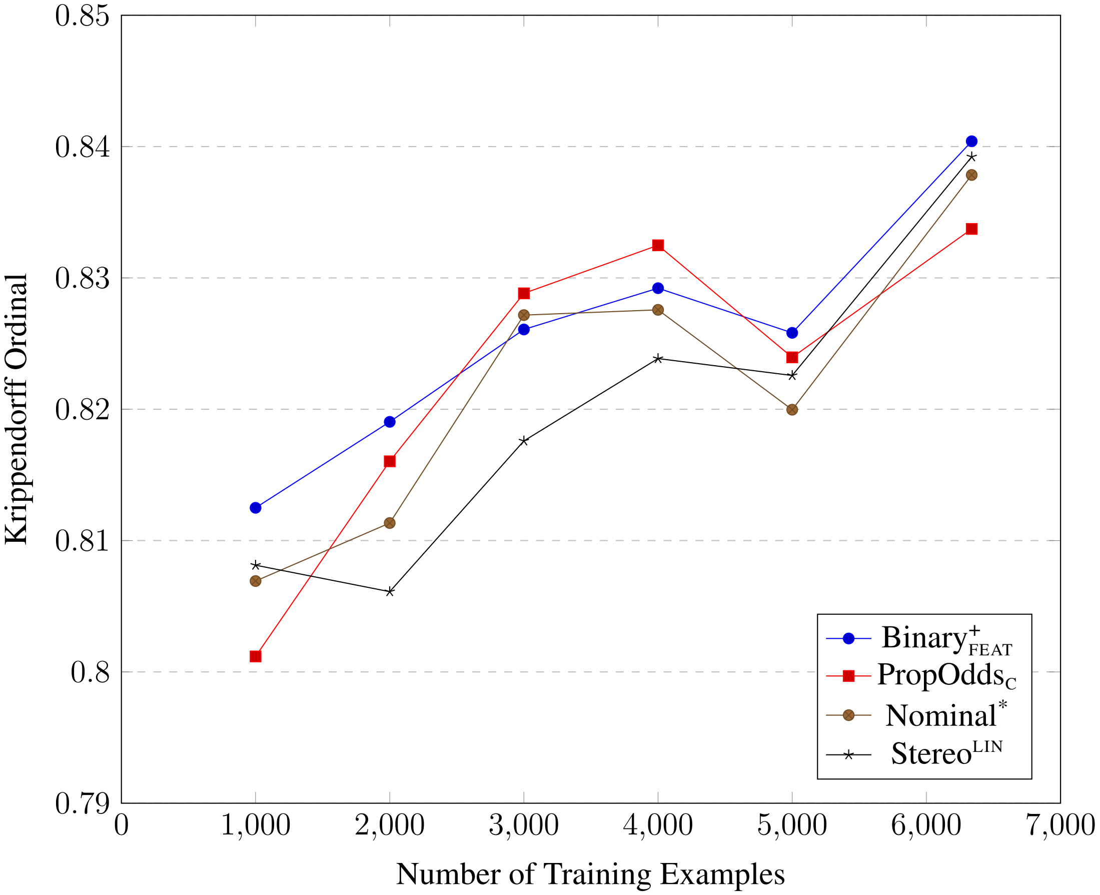


**Figure 5.8.** Clique model comparison of krippendorff-ordinal (averaged over two runs of the experiment) (evaluated on dev data) for a size-1 subset of the properties.

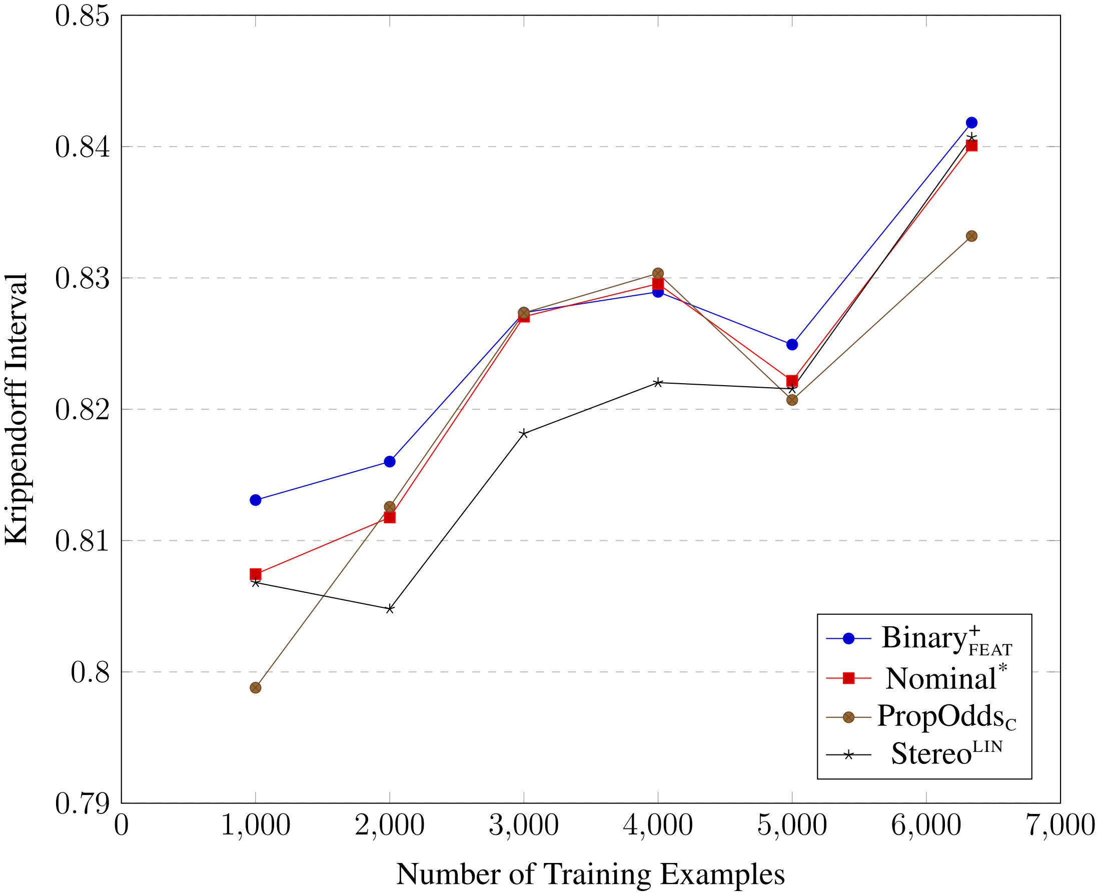


**Figure 5.9.** Clique model comparison of krippendorff-interval (averaged over two runs of the experiment) (evaluated on dev data) for a size-1 subset of the properties.

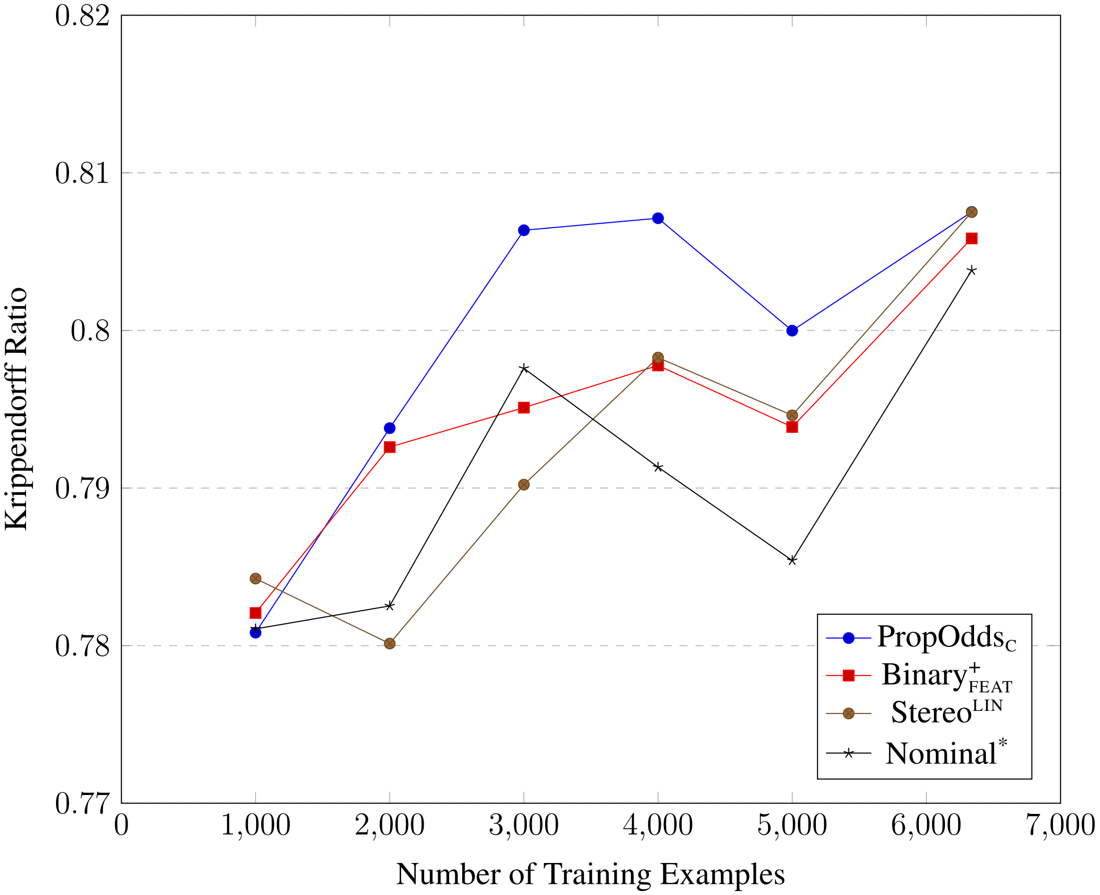


**Figure 5.10.** Clique model comparison of krippendorff-ratio (averaged over two runs of the experiment) (evaluated on dev data) for a size-1 subset of the properties.

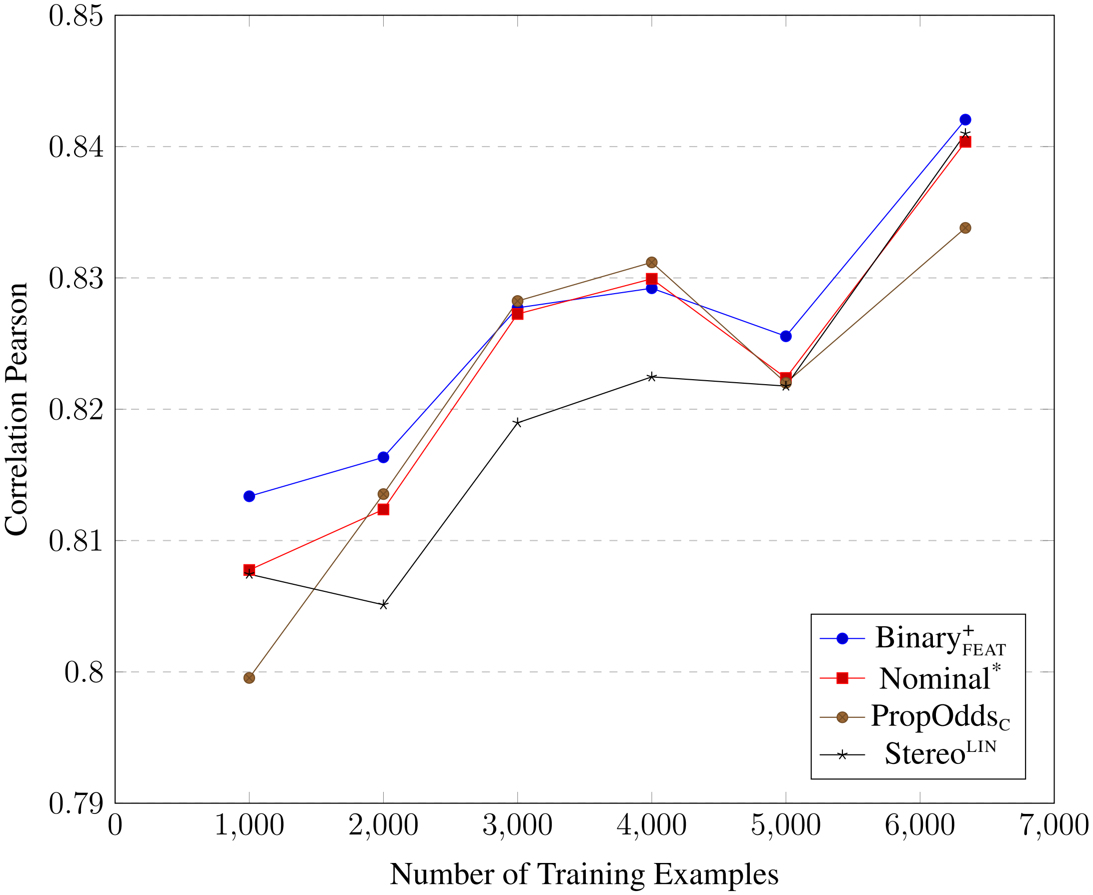


**Figure 5.11.** Clique model comparison of correlation-pearson (averaged over two runs of the experiment) (evaluated on dev data) for a size-1 subset of the properties.

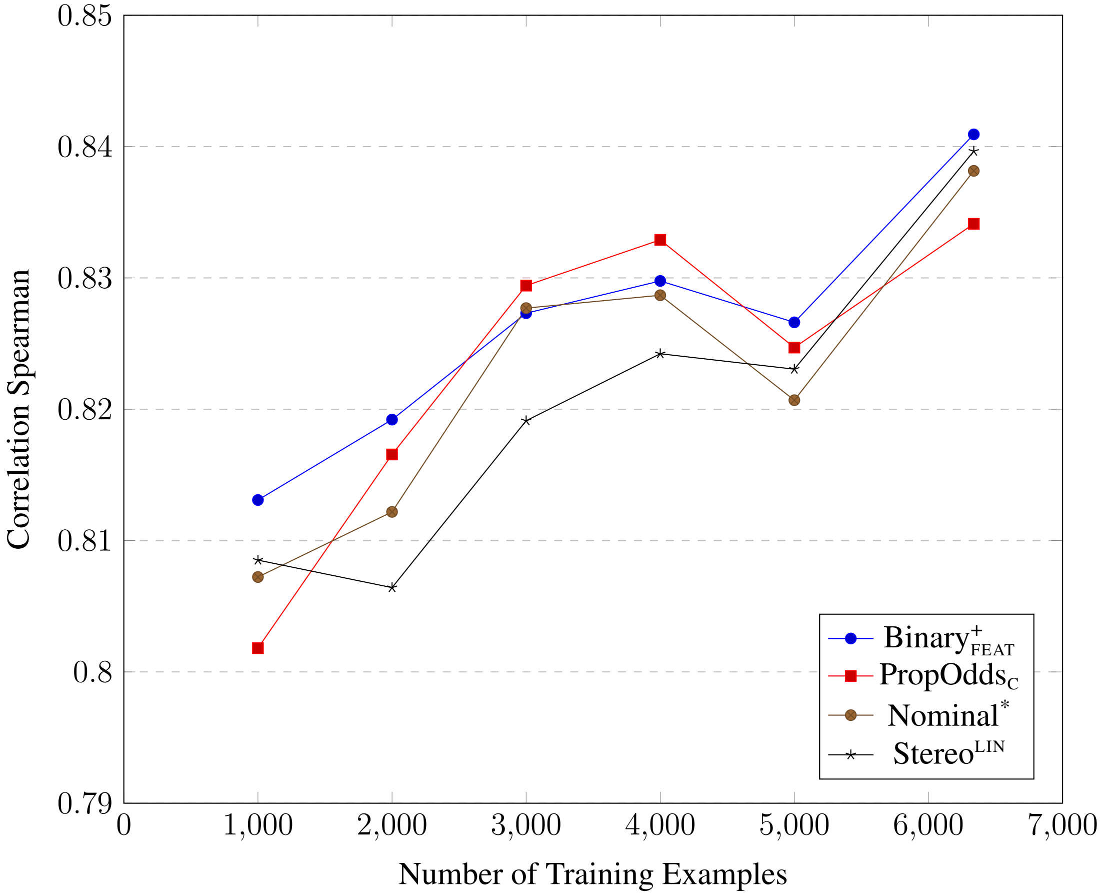


**Figure 5.12.** Clique model comparison of correlation-spearman (averaged over two runs of the experiment) (evaluated on dev data) for a size-1 subset of the properties.

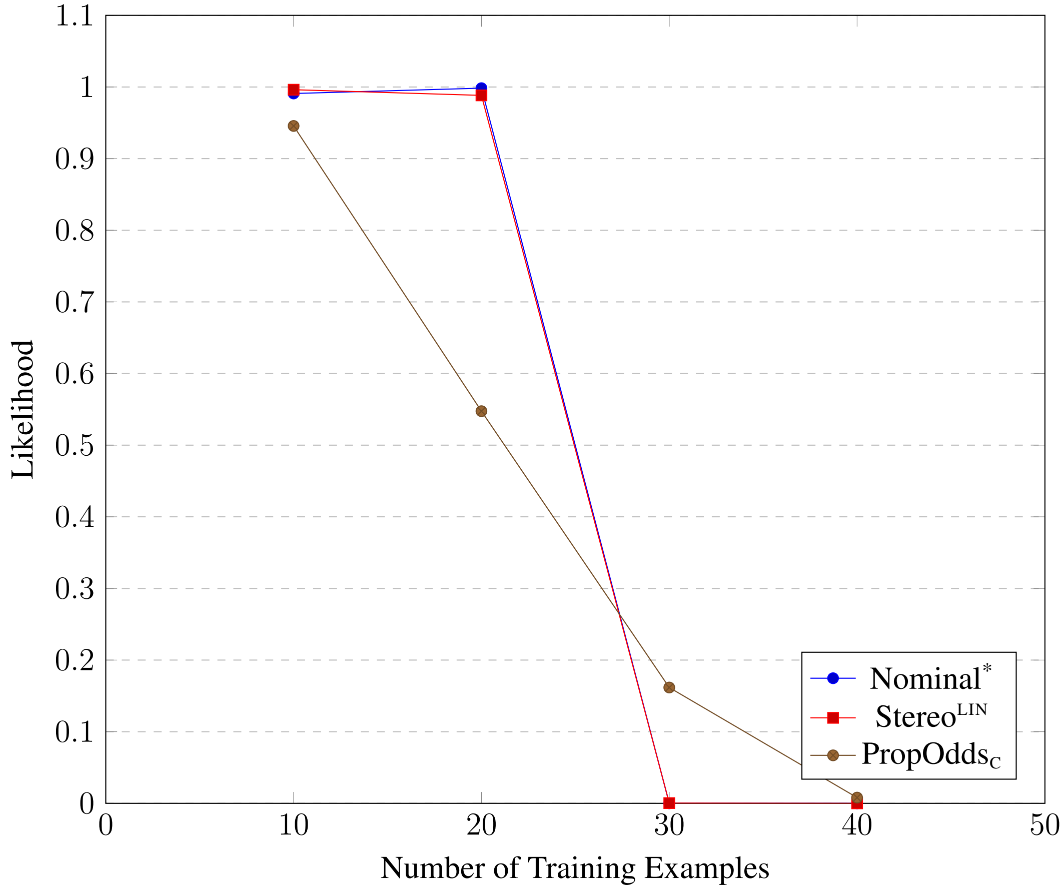


**Figure 5.13.** Clique model comparison of likelihood – tiny training sizes only (averaged over two runs of the experiment) (evaluated on train data) for a size-1 subset of the properties.

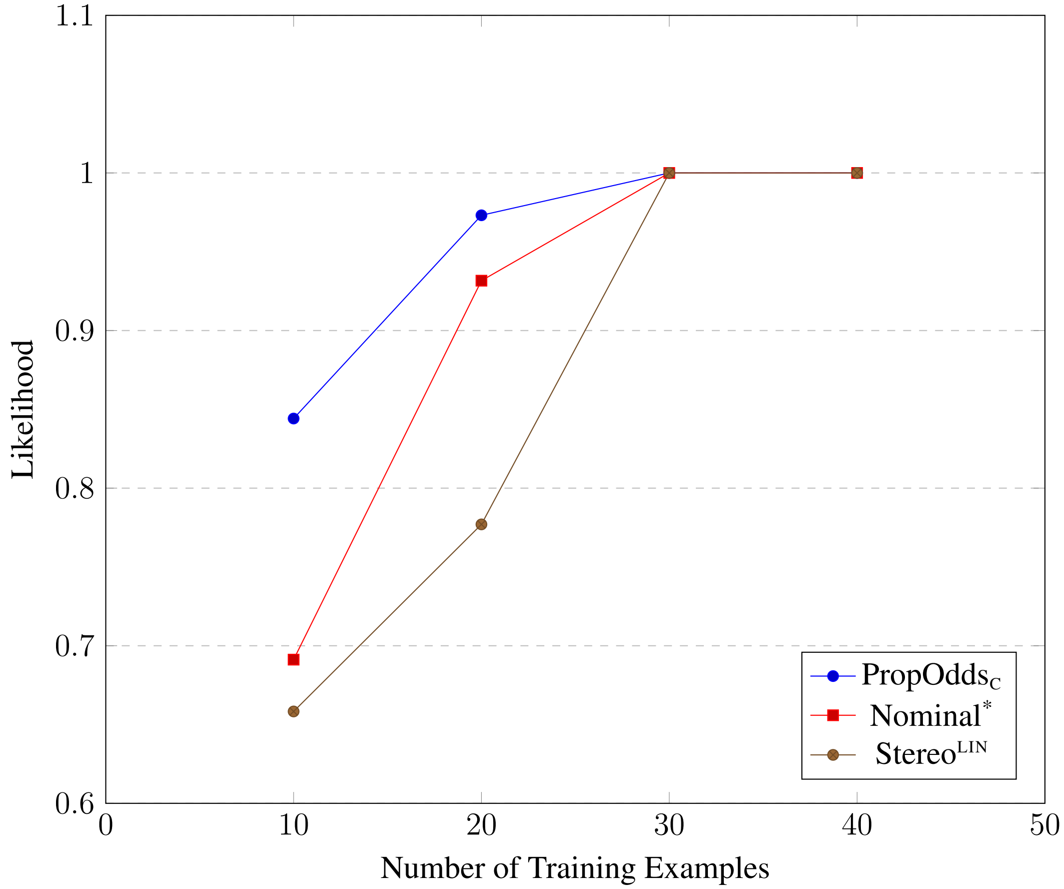


**Figure 5.14.** Clique model comparison of likelihood – tiny training sizes only (averaged over two runs of the experiment) (evaluated on val data) for a size-1 subset of the properties.

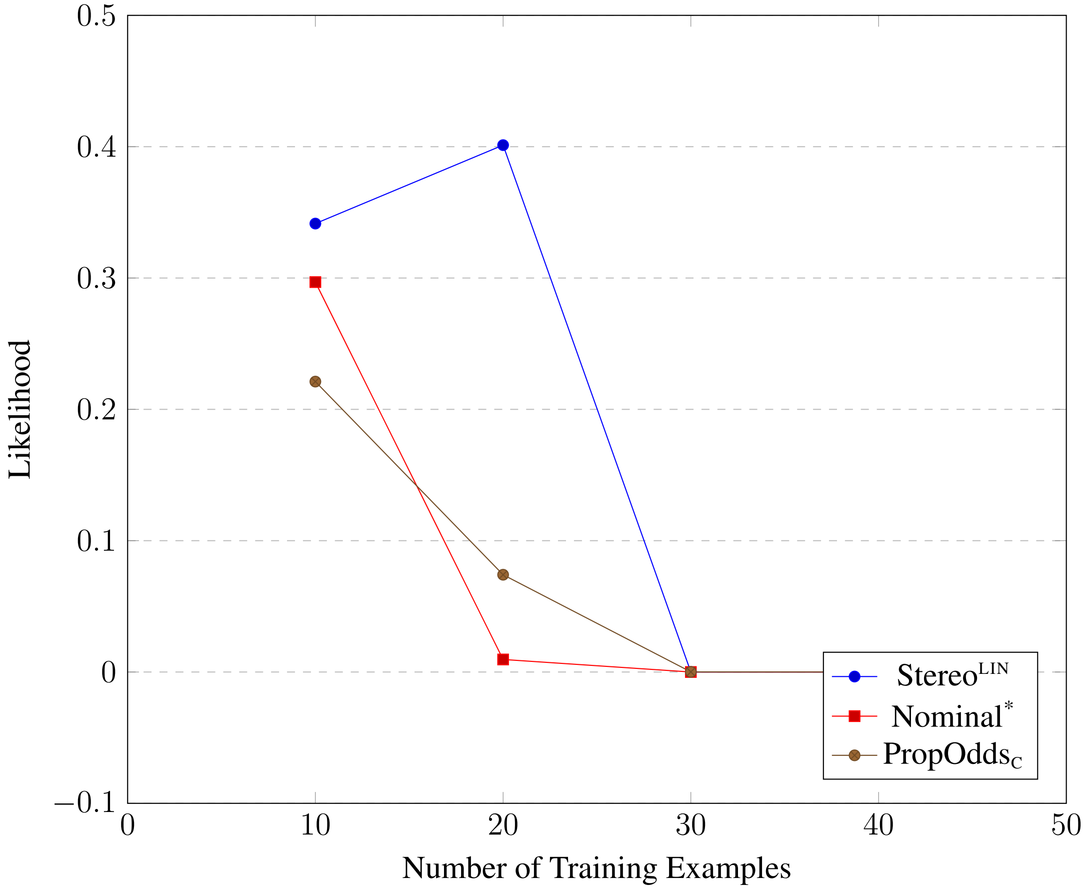


**Figure 5.15.** Clique model comparison of likelihood – tiny training sizes only (averaged over two runs of the experiment) (evaluated on dev data) for a size-1 subset of the properties.

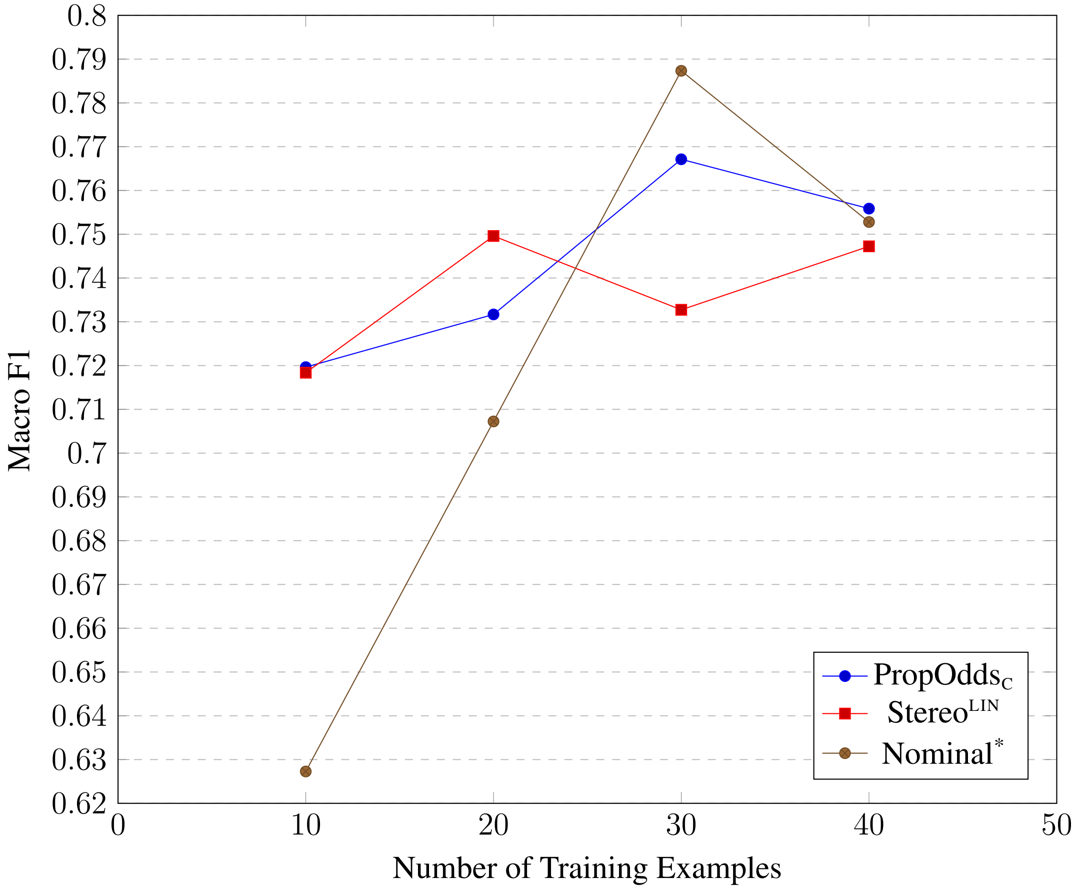


**Figure 5.16.** Clique model comparison of macro-f1 – tiny training sizes only (averaged over two runs of the experiment) (evaluated on dev data) for a size-1 subset of the properties.

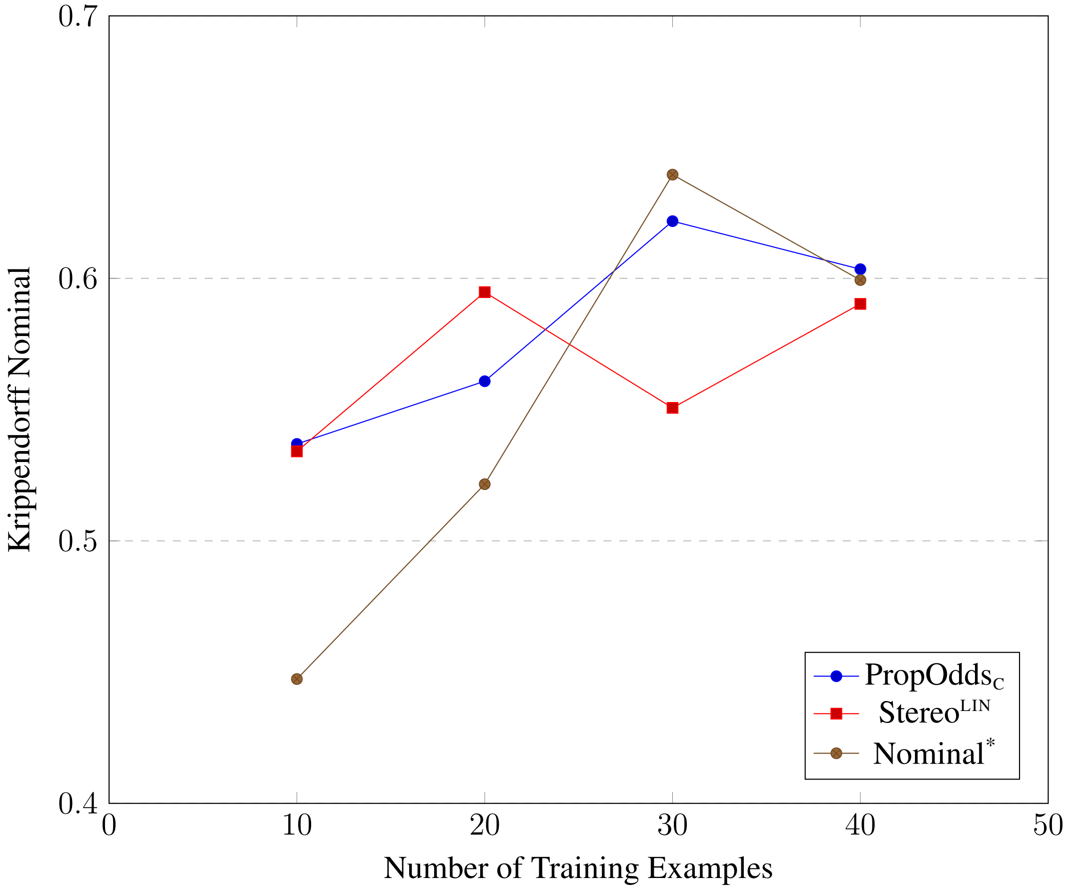


**Figure 5.17.** Clique model comparison of krippendorff-nominal – tiny training sizes only (averaged over two runs of the experiment) (evaluated on dev data) for a size-1 subset of the properties.

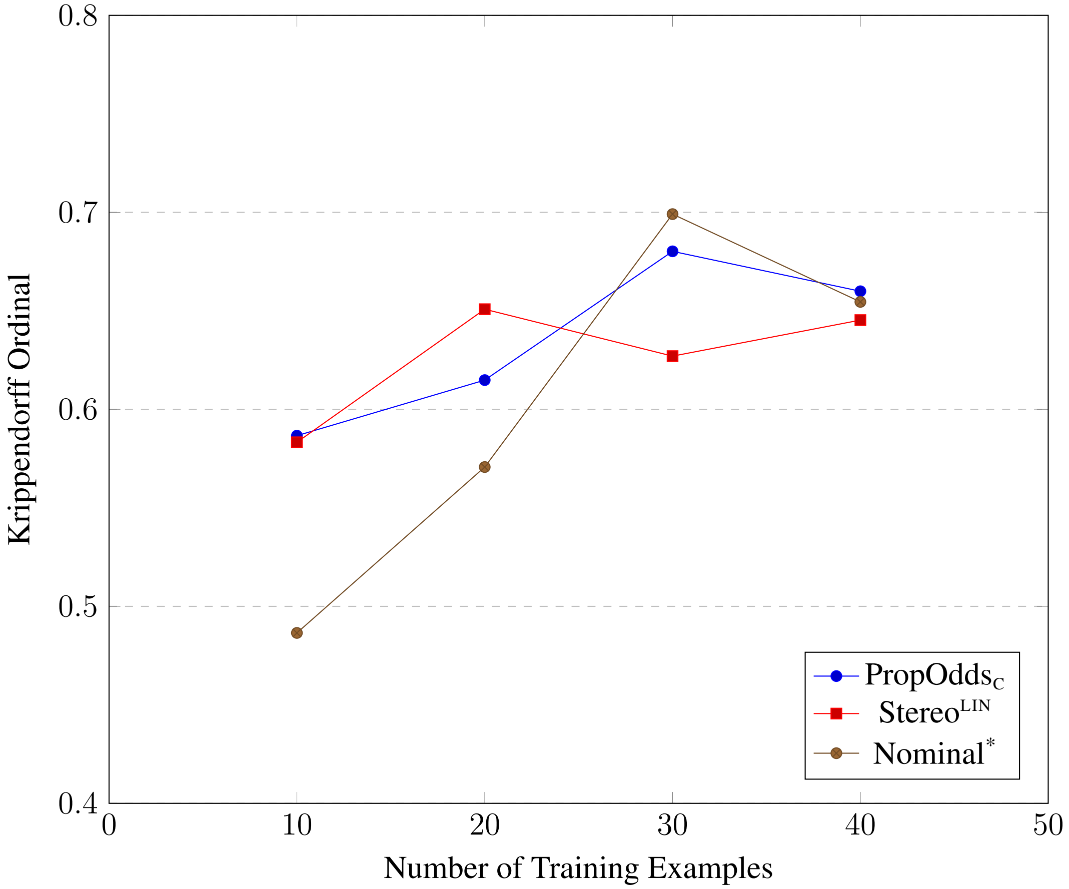


**Figure 5.18.** Clique model comparison of krippendorff-ordinal – tiny training sizes only (averaged over two runs of the experiment) (evaluated on dev data) for a size-1 subset of the properties.

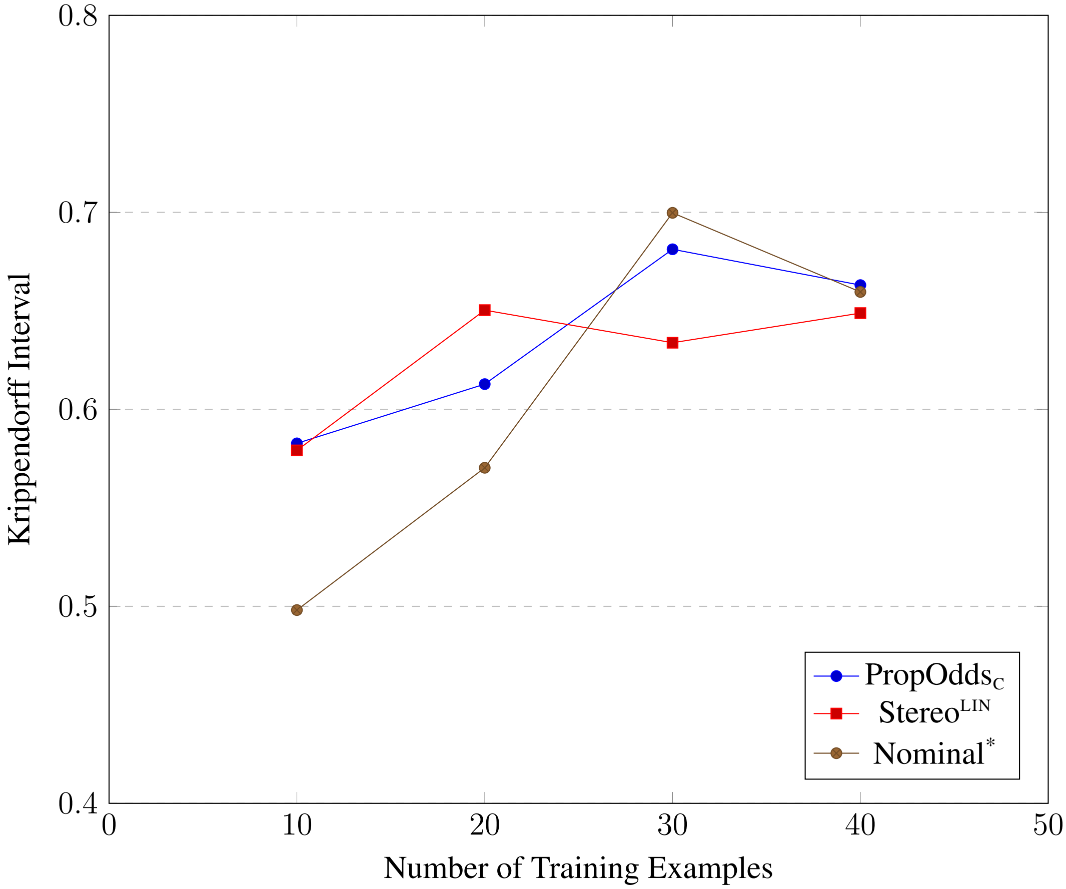


**Figure 5.19.** Clique model comparison of krippendorff-interval – tiny training sizes only (averaged over two runs of the experiment) (evaluated on dev data) for a size-1 subset of the properties.

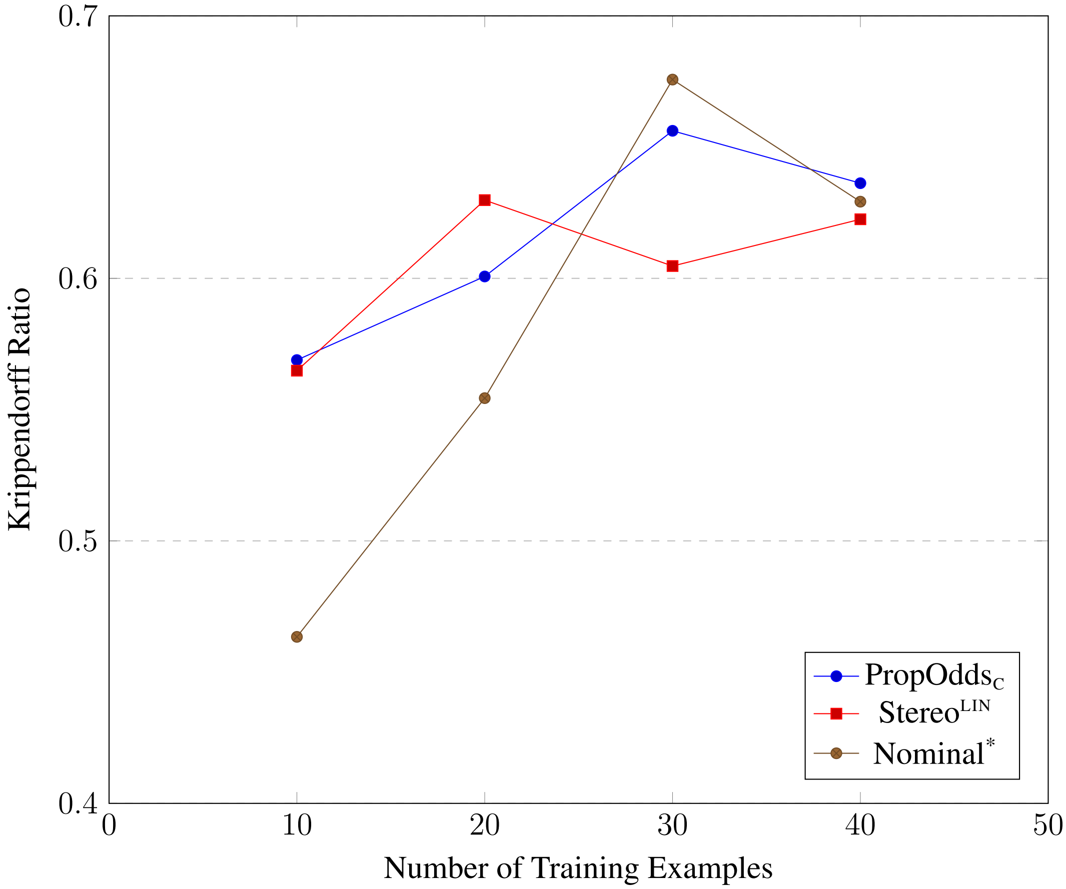


**Figure 5.20.** Clique model comparison of krippendorff-ratio – tiny training sizes only (averaged over two runs of the experiment) (evaluated on dev data) for a size-1 subset of the properties.

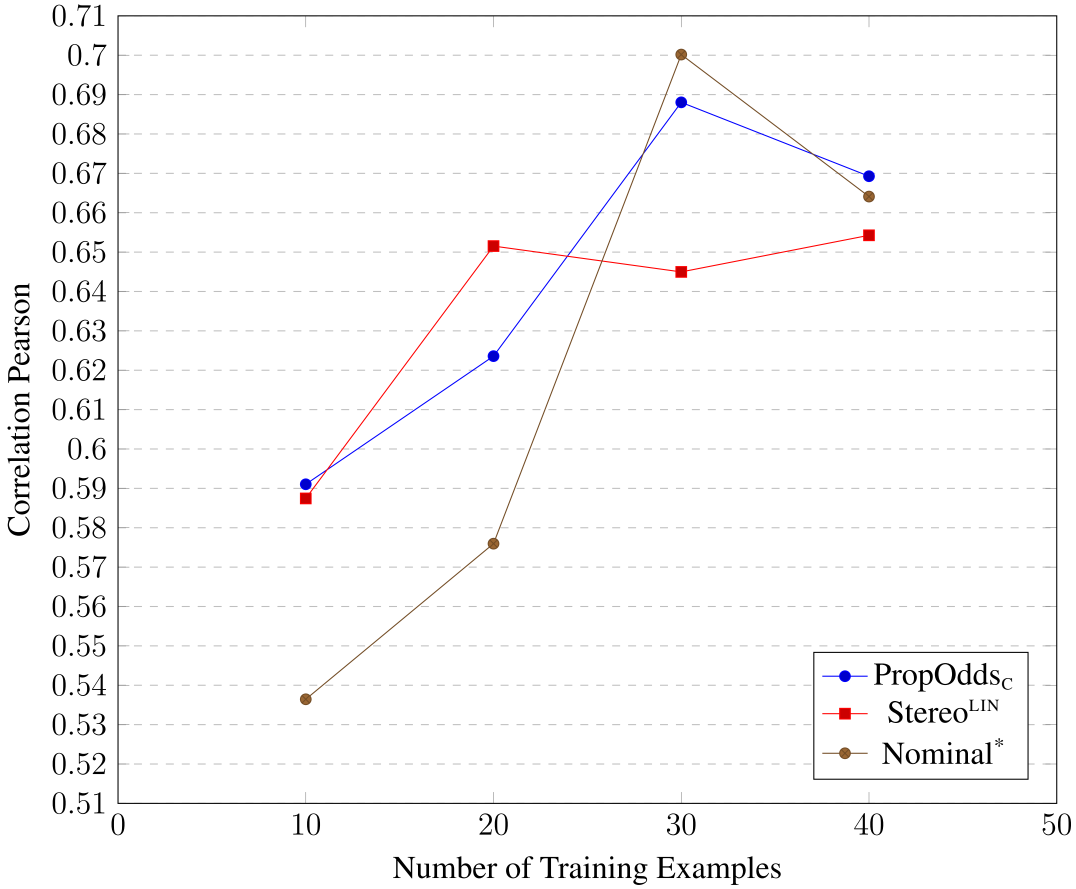


**Figure 5.21.** Clique model comparison of correlation-pearson – tiny training sizes only (averaged over two runs of the experiment) (evaluated on dev data) for a size-1 subset of the properties.

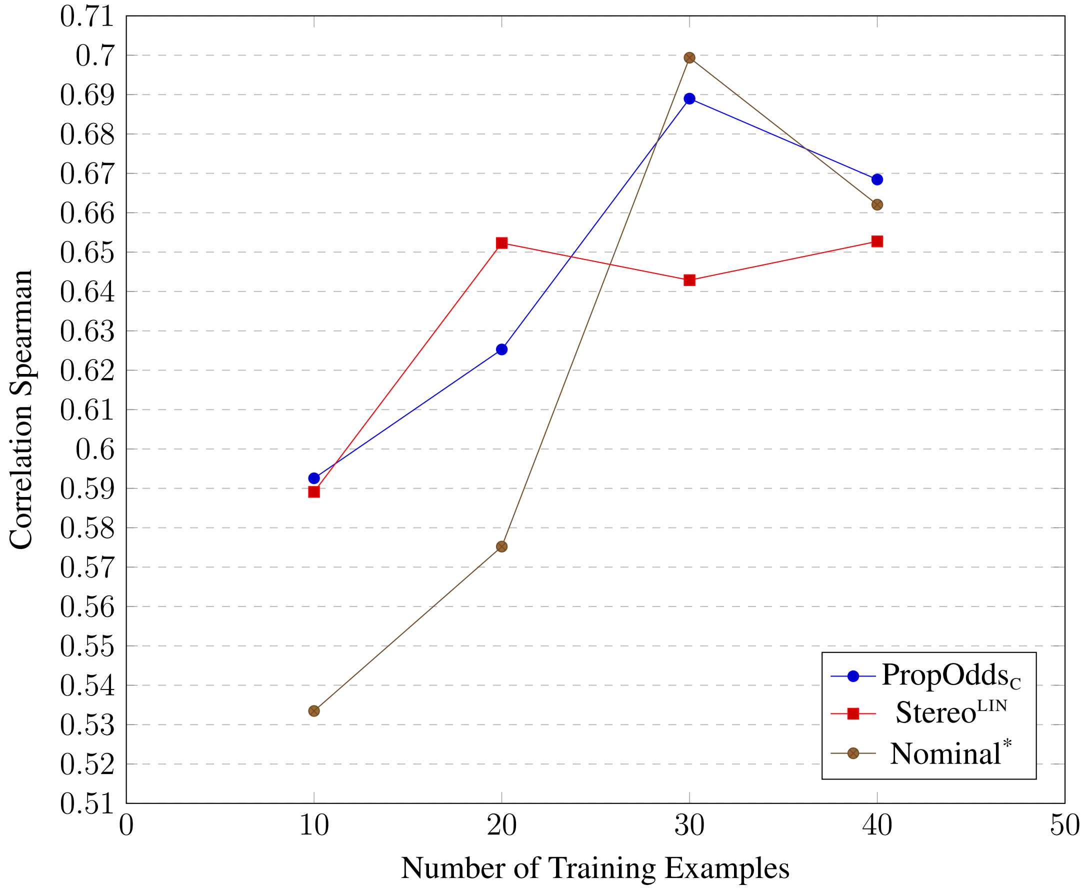


**Figure 5.22.** Clique model comparison of correlation-spearman – tiny training sizes only (averaged over two runs of the experiment) (evaluated on dev data) for a size-1 subset of the properties.

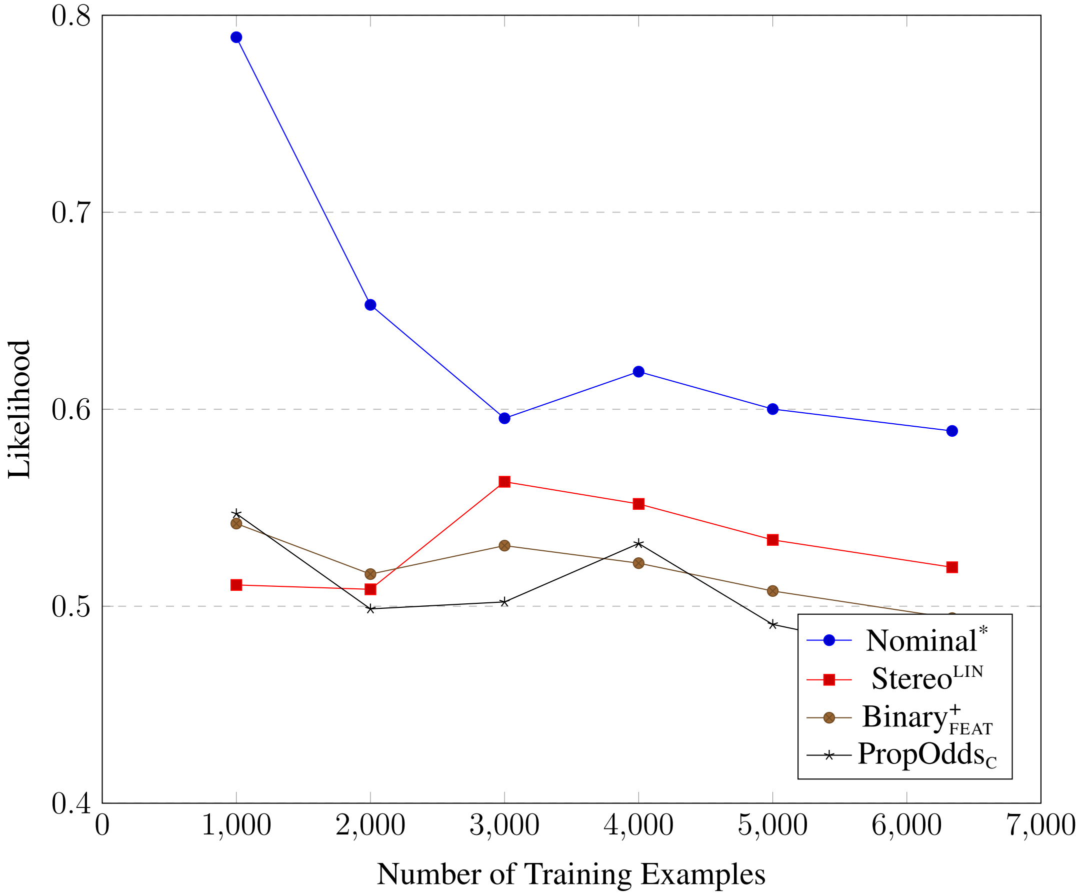


**Figure 5.23.** Clique model comparison of likelihood (evaluated on train data) for a size-3 subset of the properties.

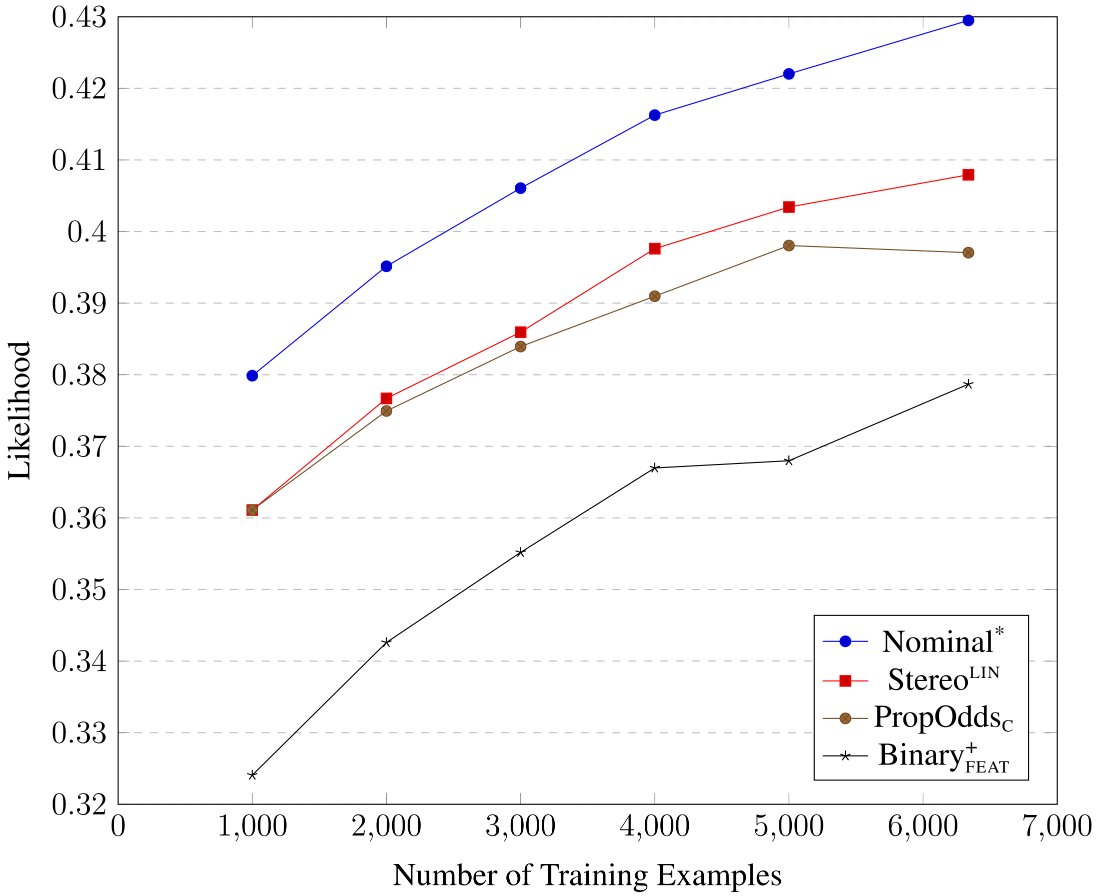


**Figure 5.24.** Clique model comparison of likelihood (evaluated on val data) for a size-3 subset of the properties.

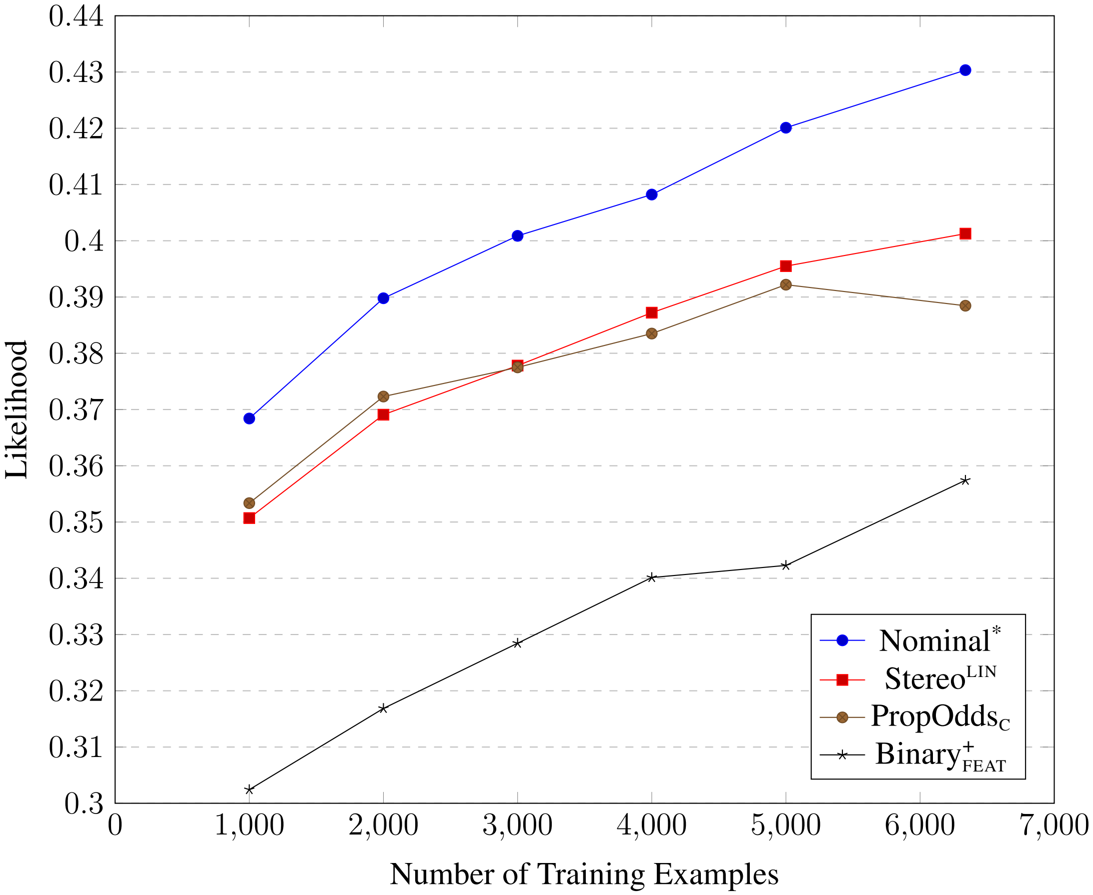


**Figure 5.25.** Clique model comparison of likelihood (evaluated on dev data) for a size-3 subset of the properties.

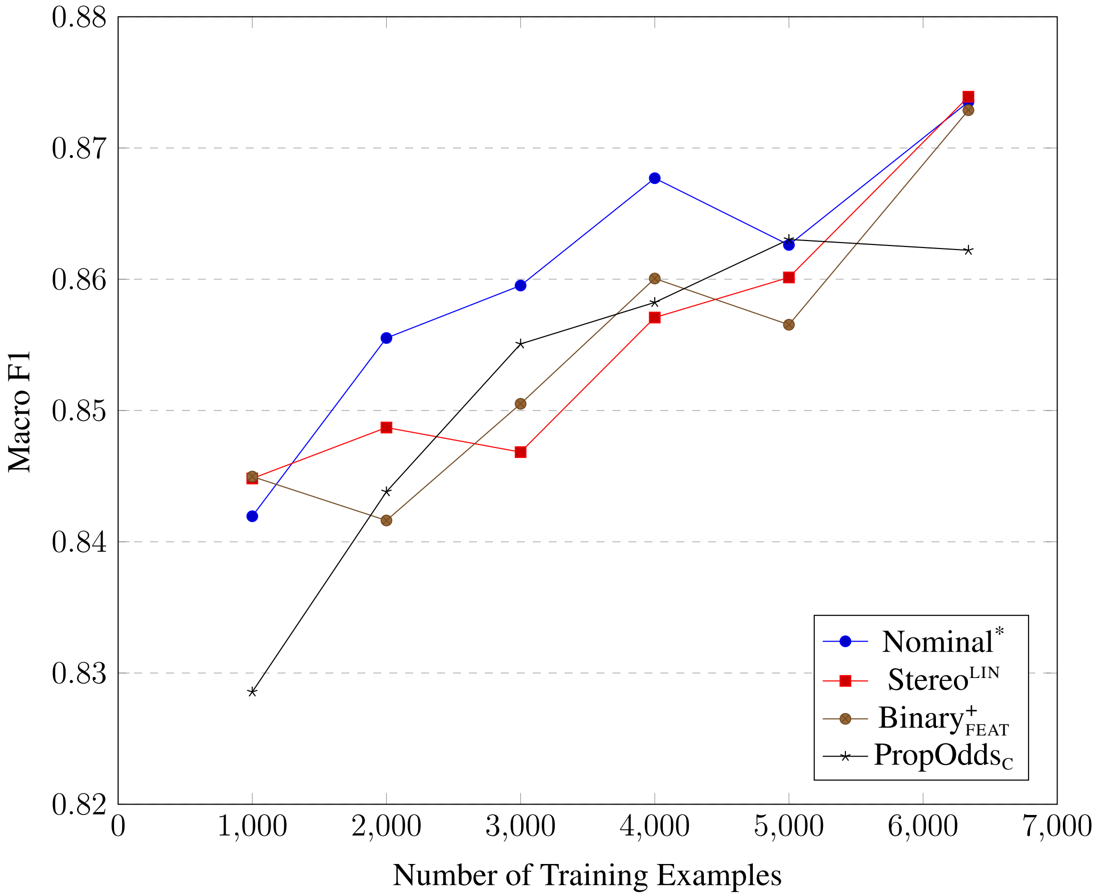


**Figure 5.26.** Clique model comparison of macro-f1 (evaluated on dev data) for a size-3 subset of the properties.

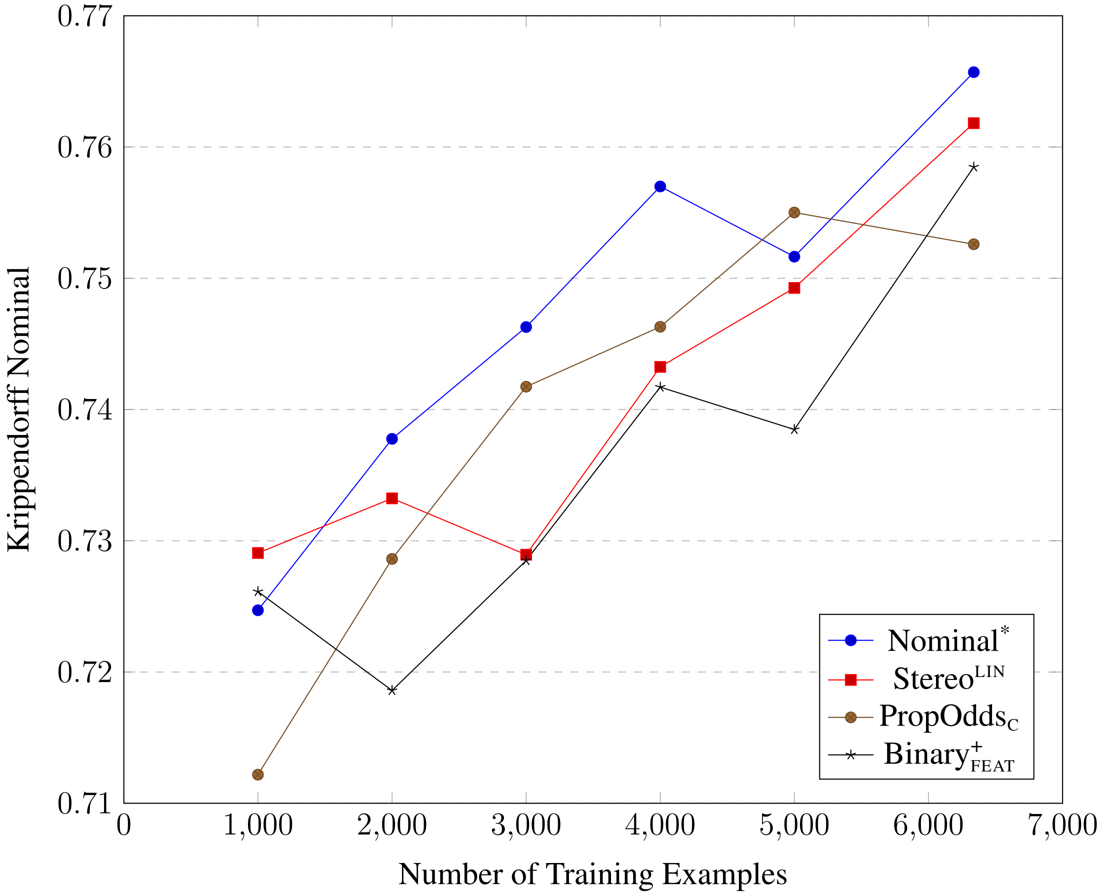


**Figure 5.27.** Clique model comparison of krippendorff-nominal (evaluated on dev data) for a size-3 subset of the properties.

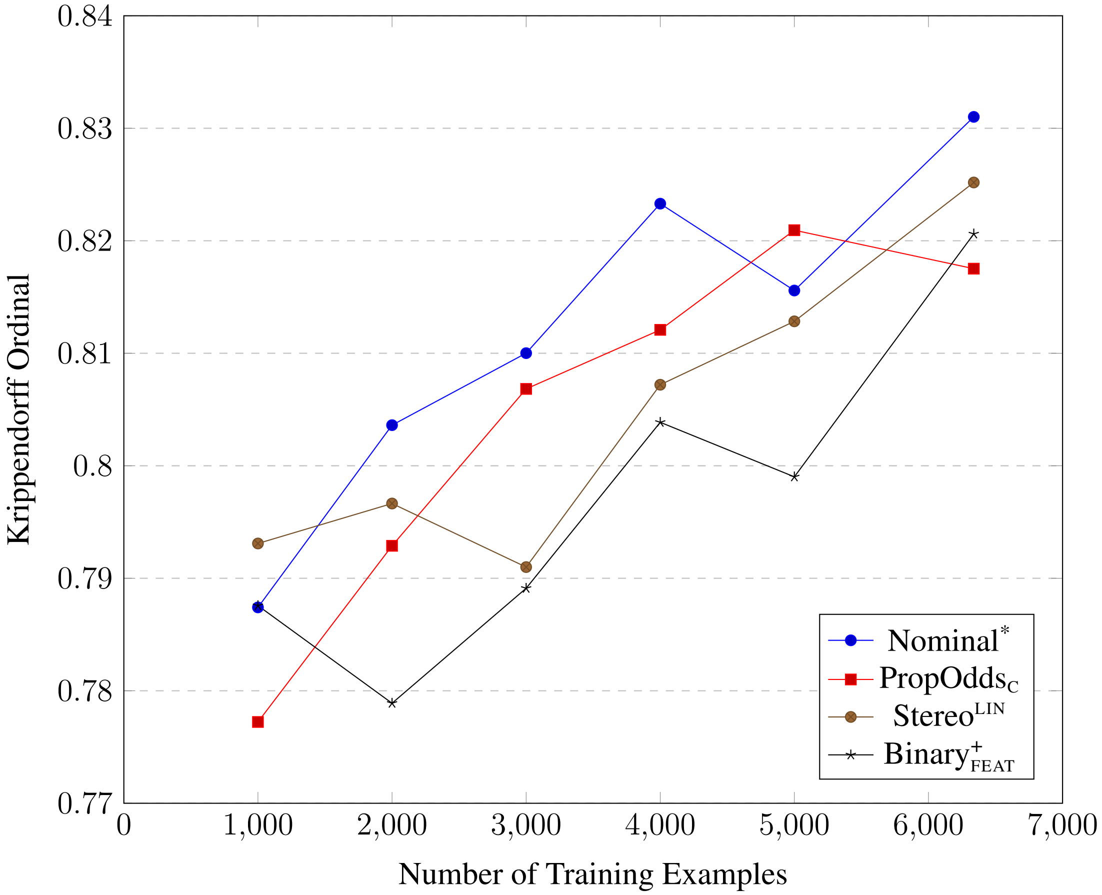


**Figure 5.28.** Clique model comparison of krippendorff-ordinal (evaluated on dev data) for a size-3 subset of the properties.

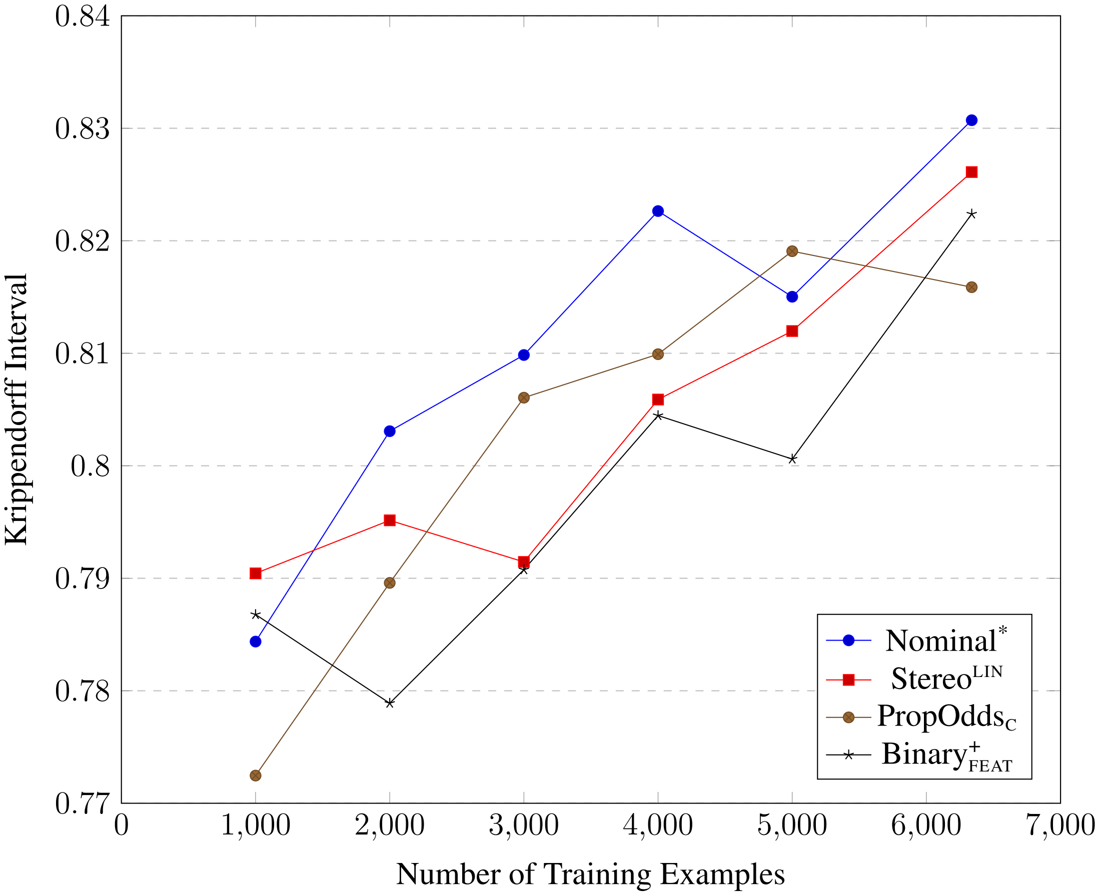


**Figure 5.29.** Clique model comparison of krippendorff-interval (evaluated on dev data) for a size-3 subset of the properties.

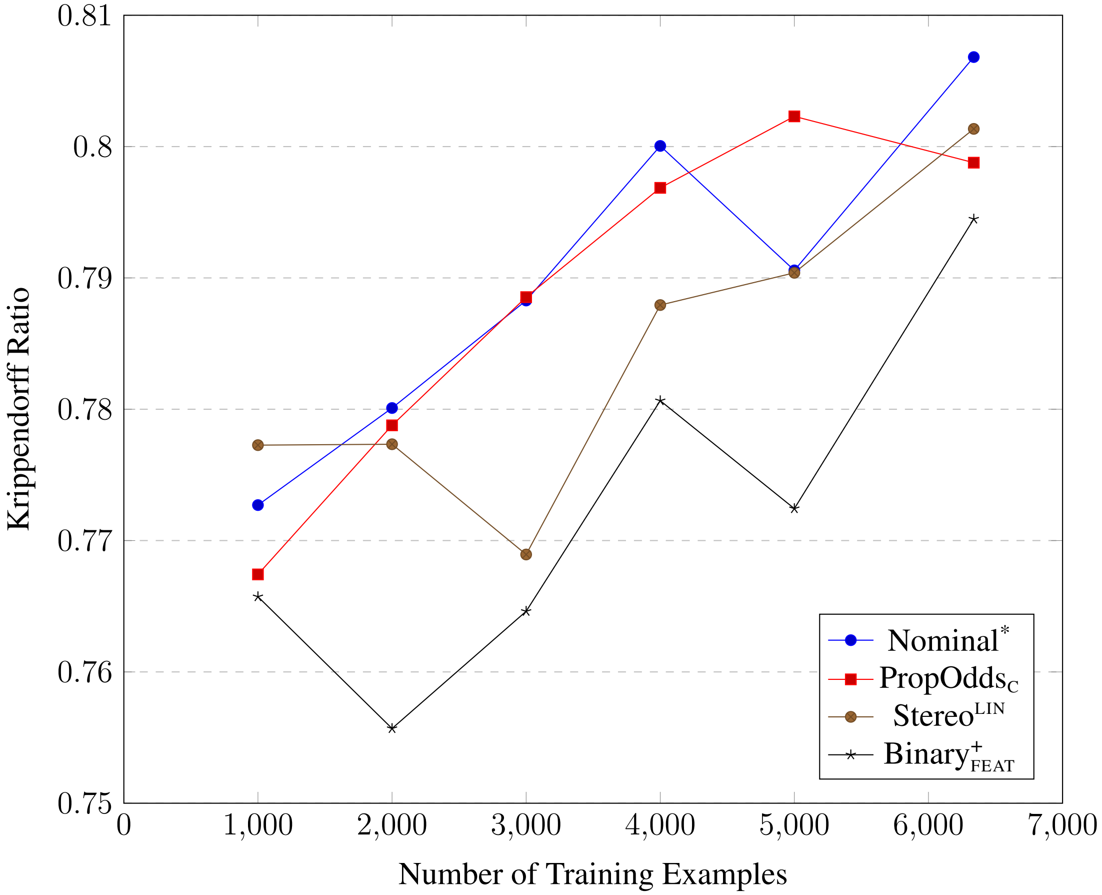


**Figure 5.30.** Clique model comparison of krippendorff-ratio (evaluated on dev data) for a size-3 subset of the properties.

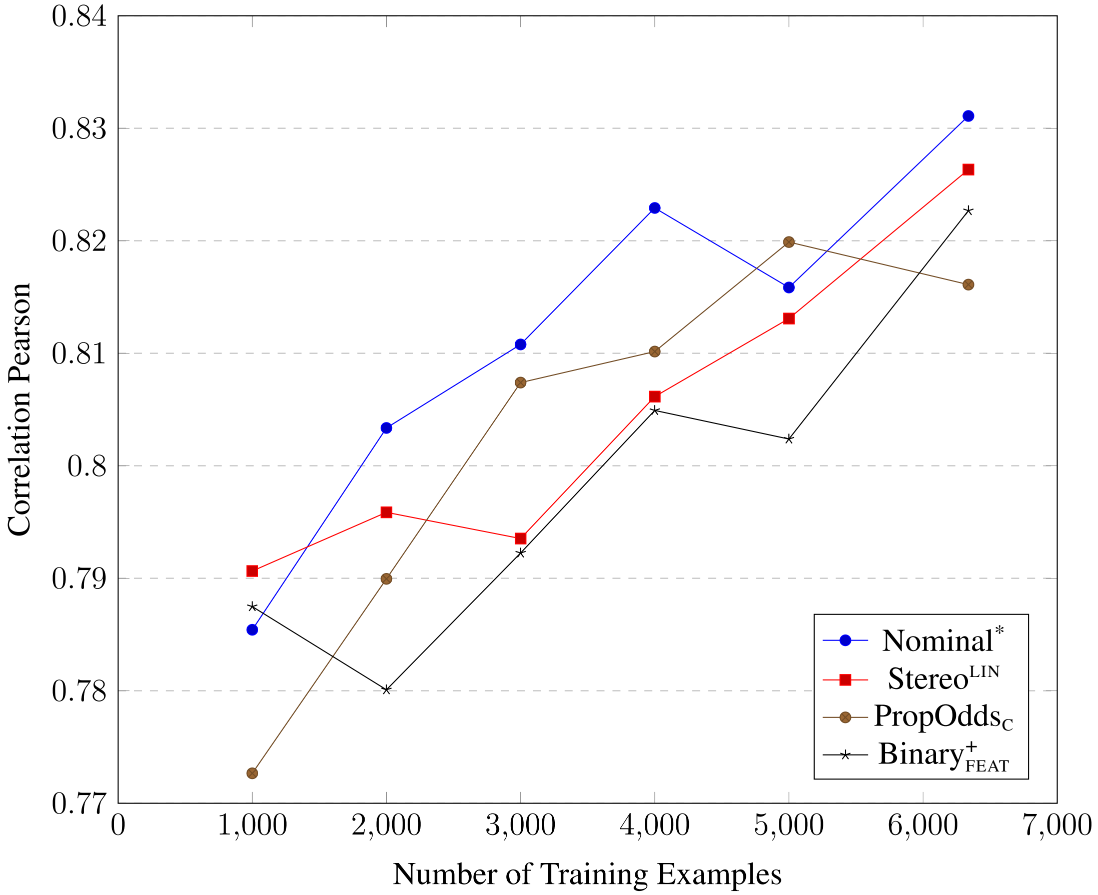


**Figure 5.31.** Clique model comparison of correlation-pearson (evaluated on dev data) for a size-3 subset of the properties.

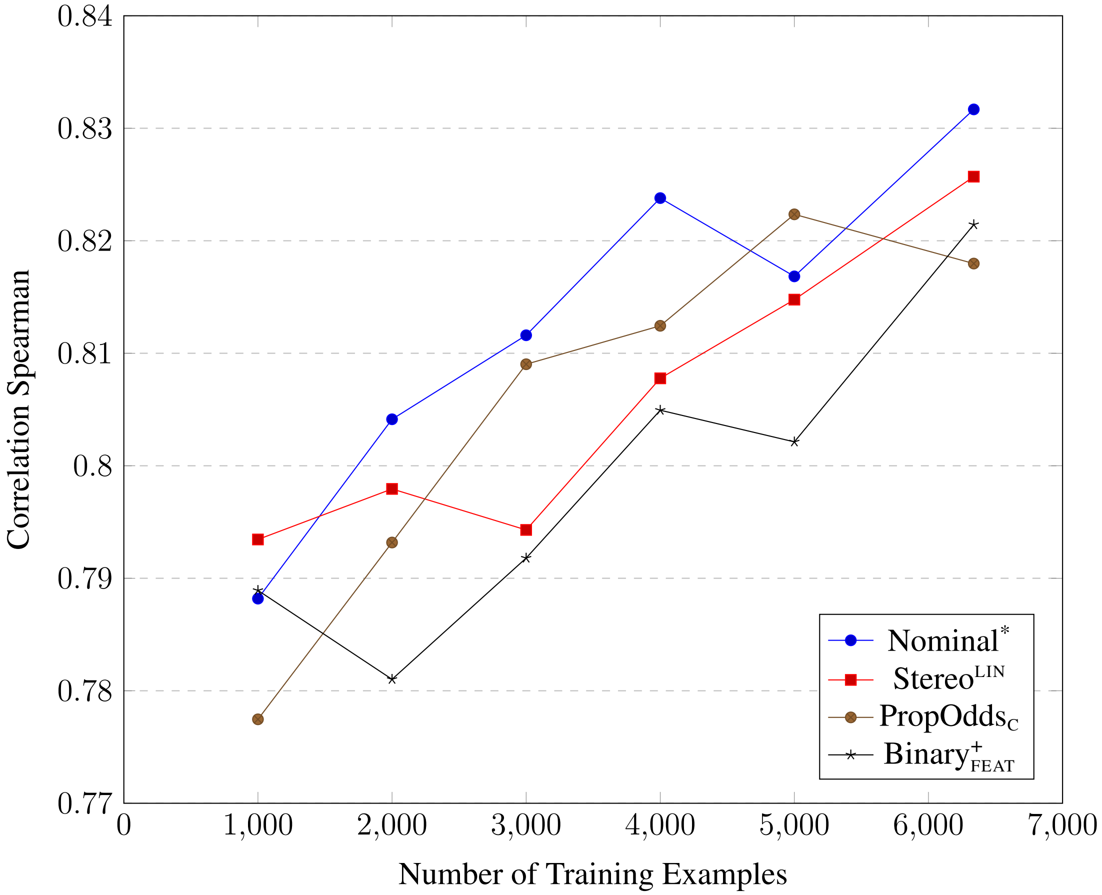


**Figure 5.32.** Clique model comparison of correlation-spearman (evaluated on dev data) for a size-3 subset of the properties.

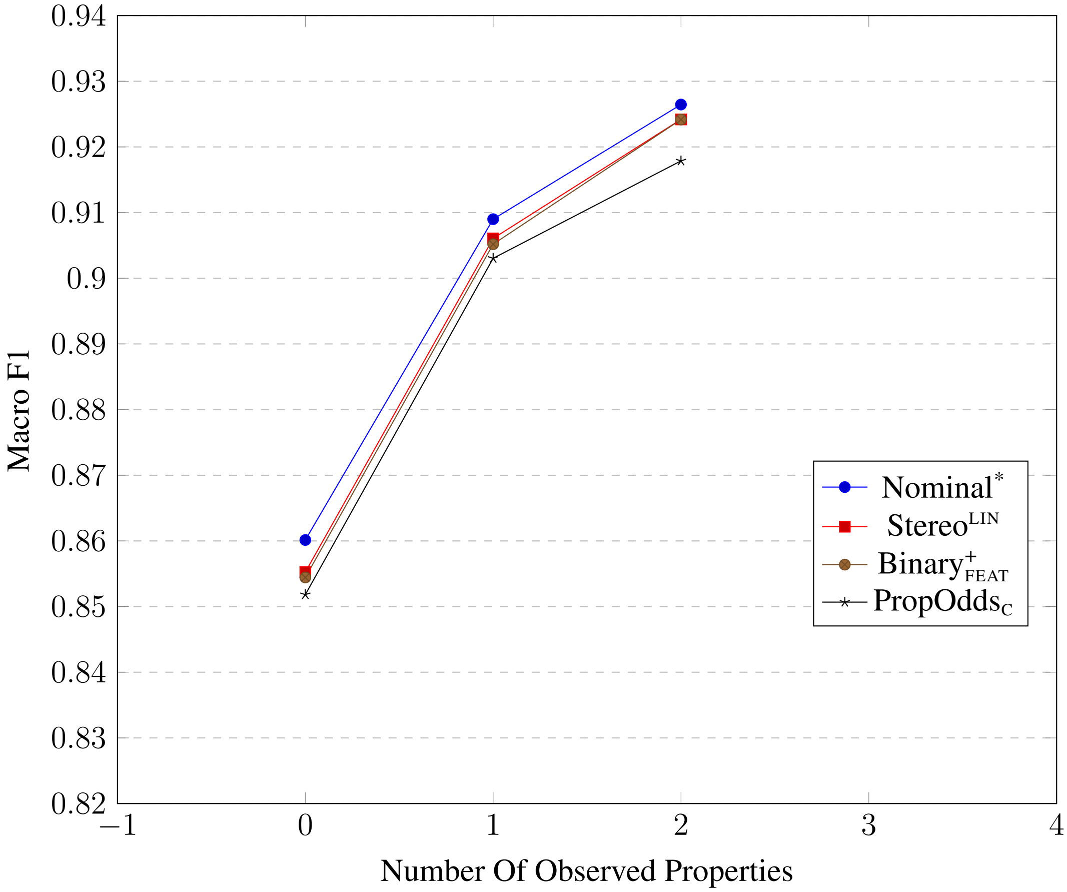


**Figure 5.33.** Conditional macro f1 (evaluated on dev data) using 3-property models

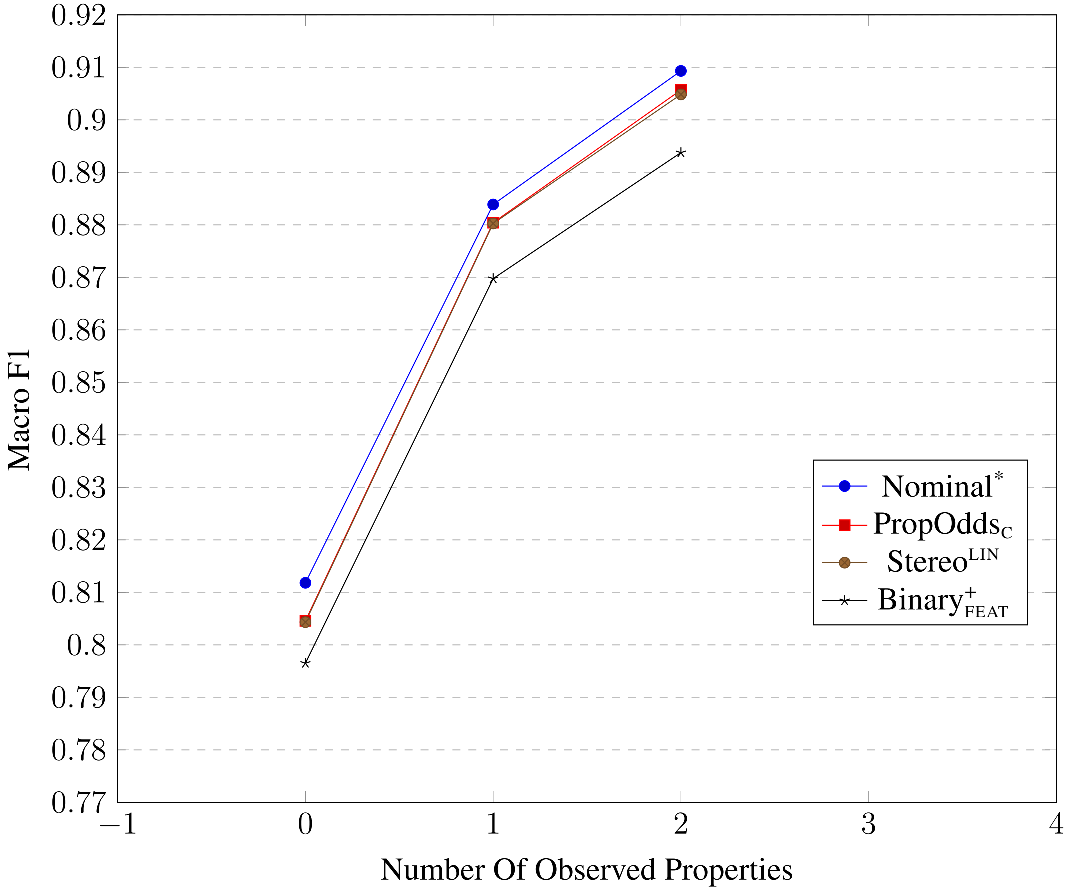


**Figure 5.34.** Conditional krippendorff-ordinal (evaluated on dev data) using 3-property models

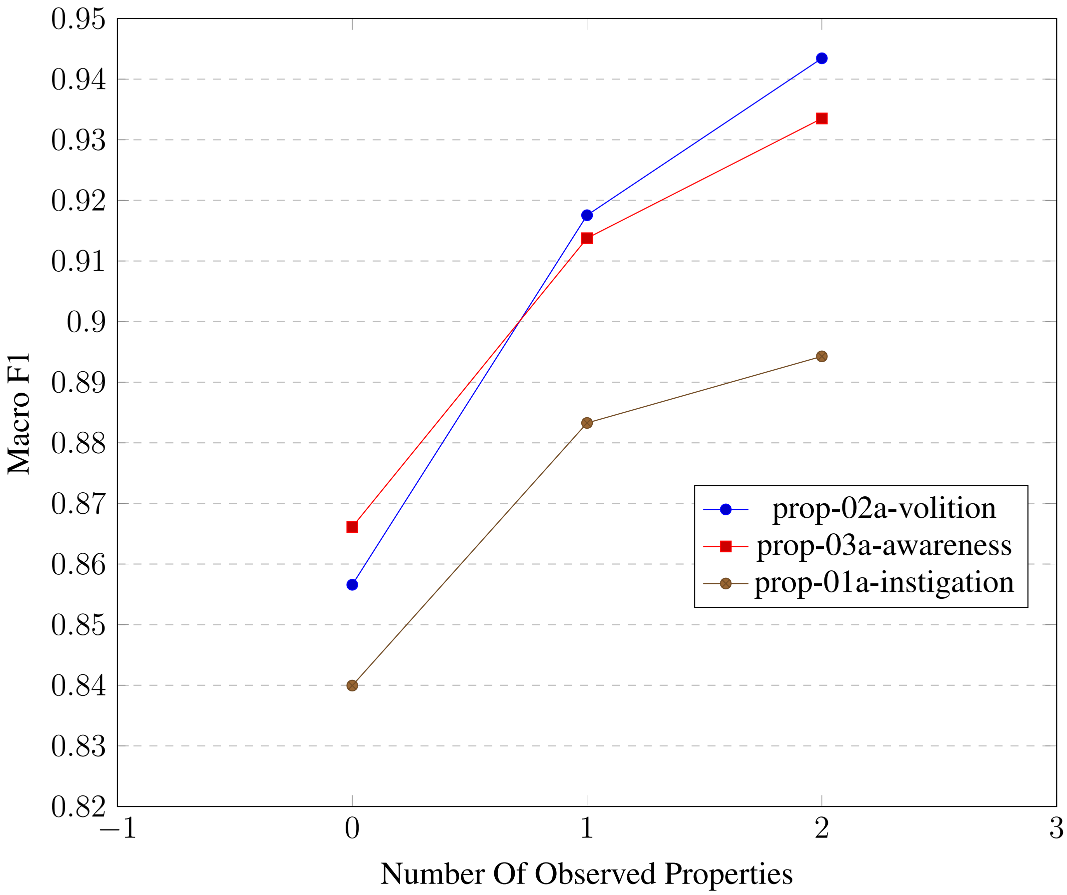


**Figure 5.35.** Property-specific conditional f1 (evaluated on dev data) using 3-property Binary$^{+}_{\text{FEAT}}$ model

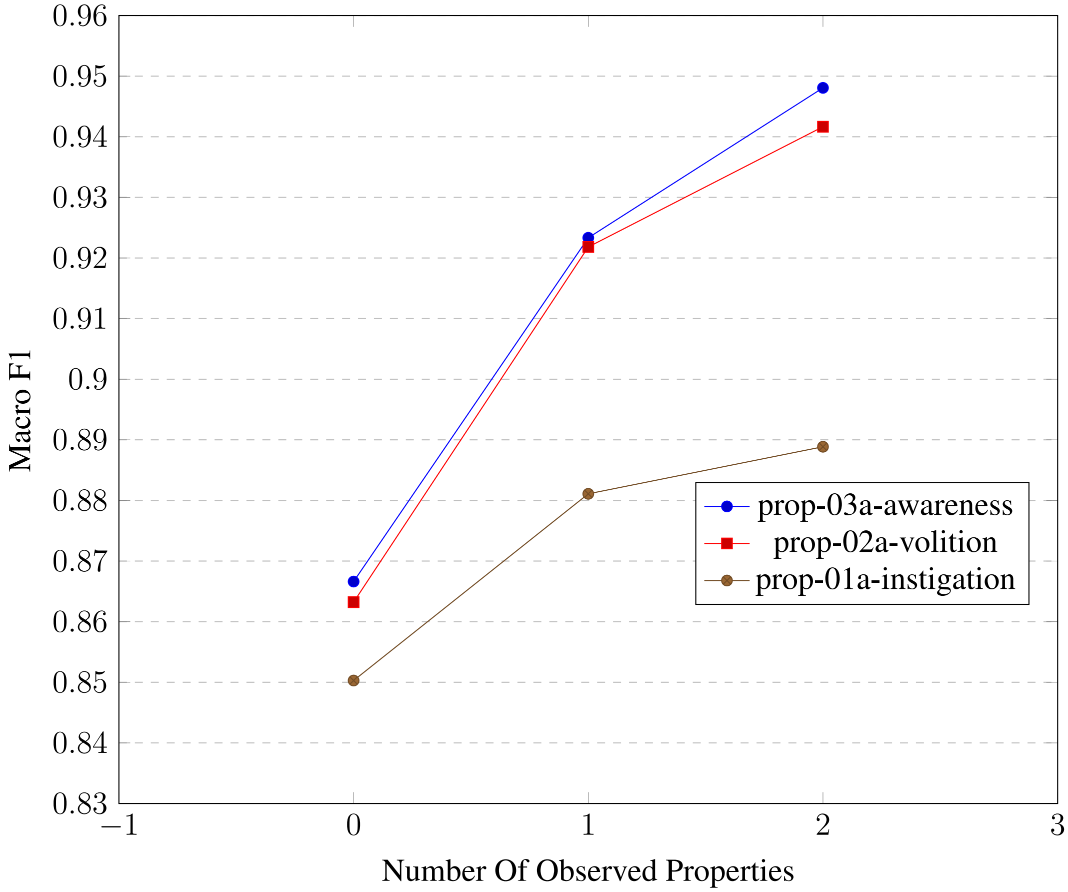


**Figure 5.36.** Property-specific conditional f1 (evaluated on dev data) using 3-property Nominal* model

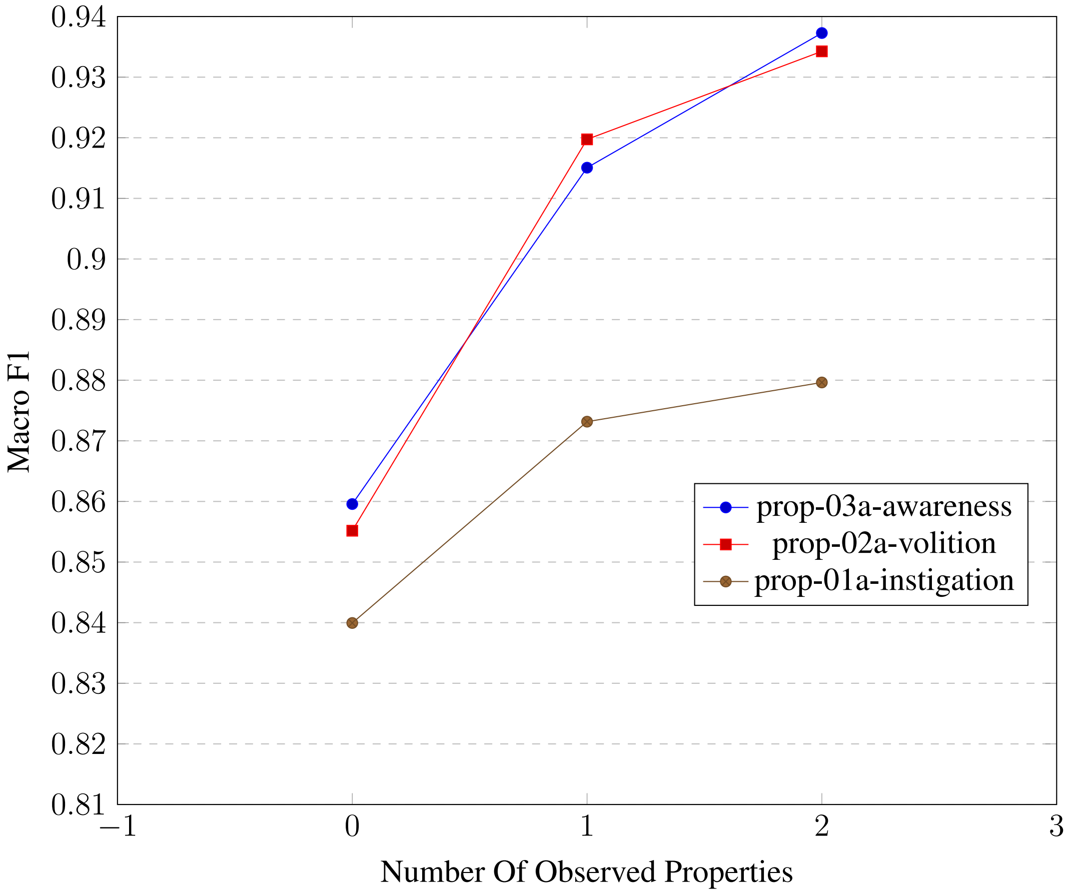


**Figure 5.37.** Property-specific conditional f1 (evaluated on dev data) using 3-property PropOdds$_{\text{C}}$ model

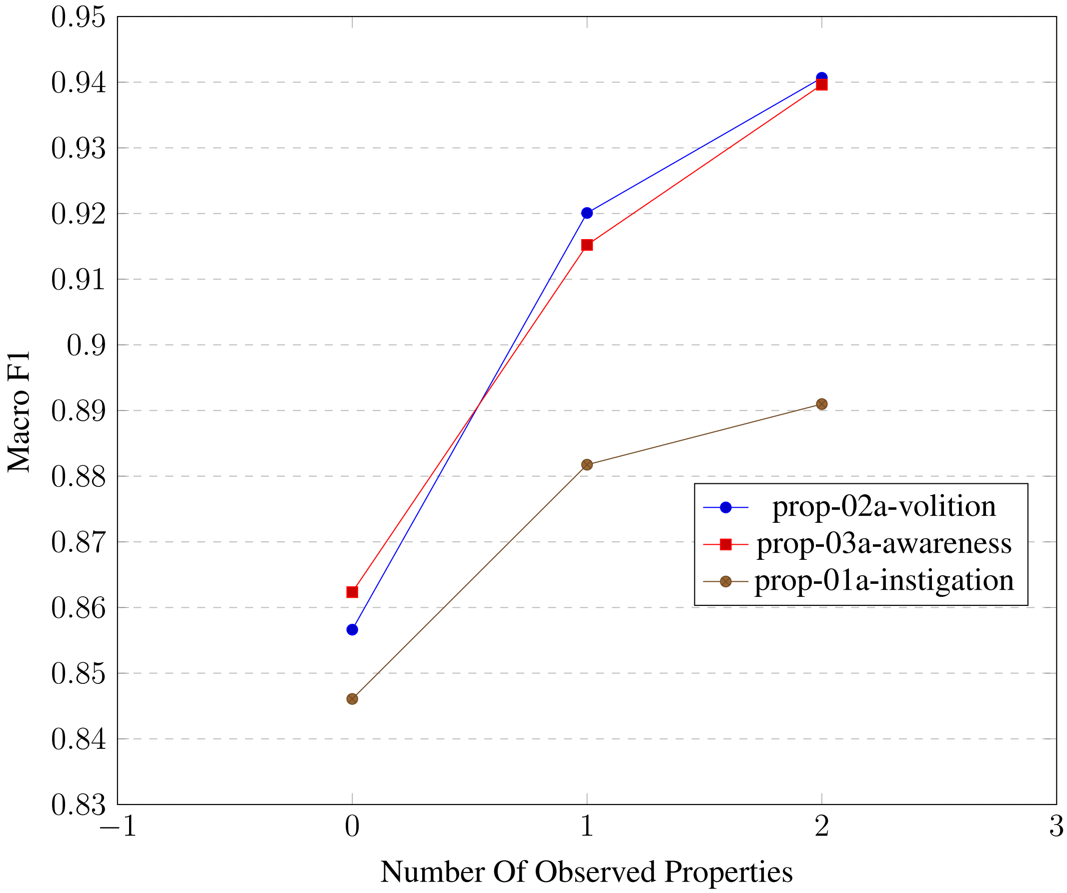


**Figure 5.38.** Property-specific conditional f1 (evaluated on dev data) using 3-property Stereo$^{\text{LIN}}$ model

Because the results were so noisy as a result of randomness in hyper-parameter selection, I repeated the experiment and averaged the two results.

Tables 5.1 and 5.2 show the comparison. Again, to reduce variance in the hope of finding trends, the results are averaged across various sizes of training data, but Figures 5.3 through 5.12 break down a subset of the models by training-set size. Figure 5.4 and 5.5 again show close correspondence between validation and held-out results. Perhaps surprisingly, the over-complete nominal model achieves even better held-out likelihood than the minimal nominal and ordinal models while being close to par with respect to other metrics. However, the ordinal approaches require much smaller models and do see some gains over the nominal model. My guess is that the reason for not having stronger gains is that the hyper parameters (learning rate, weight-decay, and batch size) together with early stopping all serve as forms of regularization that are already almost sufficient to avoid over-fitting. Thus, there is little need to explicitly restrict the expressiveness of the model— especially since regularization via dropout, etc. is not even been used here.

Since hyper-parameter selection does depend on a reasonable amount of validation data, it might be the case that extremely small amounts of training and validation data would show the benefit of the ordinal models. To investigate, I revisited the experiment on the volition property but using only 10 examples for validation and using sweeping over training data sizes of 10,20,30,40, and 50 (again, I reran the experiment twice to reduce variance). Tables 5.3 and 5.4 seem to show that the ordinal models may be useful for scenarios where resources are extremely limited. The collapsed proportional odds model, in particular, consistently outperforms most other models for these small scales. Figures 5.15 through 5.22 show the results graphically and broken down by training-set size. Figures 5.13 and

5.14 clarify why the held-out likelihood in Figure 5.15 decreases with more training—since the validation set is so limited and being used for hyperparameter selection, early-stopping seems to lead to severe under-fitting of the training data which and over-reliance on the smaller validation data.

Next, I turn to multiple-property prediction. Since the number of parameters needed grows by a factor that is exponential in the clique size and with a base equal to the effective domain size, the number of parameters needed by the nominal models grows even faster than the number needed by the ordinal models, so I expected to see clearer wins by the ordinal models when moving to pairwise models over three properties. However, the Tables 5.5 and 5.6 and Figures 5.23 through 5.32 show somewhat noisy results (I only performed a single repetition of this experiment), and it looks like the nominal model is largely on par with the ordinal models.

Finally, the same overall pattern emerges in Figures 5.33 and 5.34 where I compare the ability of the models to make use of available label observations. The over-complete nominal model essentially does as well as any model on held out data, but the ordinal models are close. The property-specific breakdowns for the top models are shown in Figures 5.35 through 5.38.

## 5.4 Related Work

Before concluding this chapter, since joint probabilistic ordinal models combine aspects from a variety of active research areas, I use this section to bring together some core insights that have contributed to the field or to my specific work in this thesis. This section is largely a stand-alone aside for the chapter, providing additional context to my work as well as exposure to and credit for ideas and possible connections emerging in the field. Raulamo-Jurvanen, Hosio, and Mäntylä (2019) also give a nice, recent survey of ordinal models.

I first review some major themes of how researchers attempt to benefit from or account for relationships among labels, citing a few examples and then highlight additional examples and approaches in Section 5.4.2.

### 5.4.1 Implications of Ordered Labels on Model and Evaluations

What do we mean when we say that the order of the labels is meaningful?

Ordinal labels might suggest constraints on the loss function—requiring that the loss of a closer label is smaller than the loss of a label that is further away. That is for true label $y$ and other labels $\hat{y}$, and $\hat{y}$, if $y < \hat{y} < \hat{y}'$, then we should have that $\forall_x, \ell(\hat{y}; y) < \ell(\hat{y}'; y)$, and, likewise, that $\forall_x \hat{y}' < \hat{y} < y \Rightarrow \ell(\hat{y}; y) < \ell(\hat{y}'; y)$. Again, although possibly a reasonable restriction, it does not strictly follow from recognizing that labels have a meaningful order. However, such a constraint would allow one labeling to be said to "dominate" another labeling with respect to *any* constrained objective function if the former predicts labels that are at least as "close" as the competitor for all examples in the dataset.

An alternative constraint on the loss function is that it must be able to be captured in terms of a cost matrix $\mathcal{C} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. For example, zero-one loss can be captured by the identity matrix. In other words, the assumption is that we can replace the instance-specific cost function $\ell_x$ with some global cost function $\ell(\hat{y}; y) = \mathcal{C}_{y;\hat{y}}$. Although even nominal tasks may employ such generalized cost function constraints, some authors propose that ordinal tasks will likely put additional constraints on $\mathcal{C}$. For example, in H.-T. Lin (2008) (see also Li and H.-t. Lin (2007)), many of their theoretic results regarding reductions from ordinal to binary classification depend on so called *V-shaped rows*. However, such a matrix formulation may not capture (or even be an adequate approximation) of every ordinal objective, and popular cost matrices like those suggested in P. A. Gutiérrez et al. (2016) are even less likely to fit a given situation. Nevertheless, if your labels are ordinal, it is likely that a non-identity matrix formulation will be more appropriate than an identity matrix formulation.

Some argue that ordinal labels suggest a constraint on the posterior distribution over labels assigned by a model given an input, e.g. that the distribution should be unimodal (Beckham and Pal, 2017). While this may (or may not) be a useful bias, it does not strictly follow from the assumption of ordinality. For example, in the case of survey questions where respondents are encouraged to express certainty whenever possible, a non-unimodal prior distribution that expects polar responses is likely well justified when there are not sufficient input features to disambiguate between examples that are strongly in favor vs strongly against. Some approaches formulate ordinal regression in an SVM-style framework, using the ordinal labels to define margin constraints.

Even if the suspected ordinality does not unequivocally warrant categorical constraints on the posterior of the model, nevertheless it may suggest model constraints in order to achieve a more parsimonious (and therefore statistically powerful) model. Some argue that such distinctions should be made on an empirical bases in light of available features and labeled data. Specifically, the posthumously published work of Anderson (1984),[7] proposes using the multinomial logistic regression framework so that one can statistically answer questions about the nature of the relationship between input features $x$ and output labels $y$. For example, what is the dimensionality of the relationship (i.e. can the labels be predicted from a single linear function of $x$ or are more warranted)? Are adjacent labels statistically indistinguishable under the model or should they be combined for the purpose of modeling (i.e. is there a statistically significant difference in the estimate for coefficients representing two different labels)? If the relationship is one-dimensional, are the labels actually monotonic with respect to the latent scalar? Such an open-ended view of ordinal regression was supported by the findings of C. Greenwood and Farewell (1988) who found that, despite a reasonable case for arguing that their dependent variable was "ordinal", their analysis favored the two dimension model of Anderson over the respective one-dimensional models of McCullagh (1980) and Fienberg (1980). Anderson further identifies that there may be important distinctions between what he calls "grouped continuous" ordinal labels which are scalar quantities that are binned and "assessed" (or "judged") ordinal labels which are responses natively given on an ordinal scale.

Torra et al. (2006) point out that one cost of treating ordinal problems as nominal is a loss of efficiency; i.e. possibly estimating more parameters than necessary and therefore risking non-significance of results. They learn a generic mapping from each label to a point on the $[0, 1]$ interval and then perform ordinal regression by way of linear regression using that label projection.

[7] In fact, the work was read before the Royal Statistical Society by none other than R. L. Plackett (known for the Placket-Luce Model (Plackett, 1975)).

P. A. Gutiérrez et al. (2016) cite an array of work from various fields where an ordinal model is able to outperform the nominal alternative. Pedro Antonio Gutiérrez et al. (2012) review some early approaches to ordinal regression and specifically look at the correlation between a number of measures for evaluating ordinal regression: accuracy, micro averaged absolute error (MAE), macro averaged absolute error (AMAE), max average absolute error (MMAE), and Kendall's tau. Despite a careful experimental setup with appropriate attention to hyper-parameter selection, the results were quite scattered among the proposed methods, depending on the particular dataset. The authors suggest that the MMAE metric is potentially interesting in that it correlates least with the other measures, and therefore they use it to make final assessment. No further motivation for this choice is given.

While it is easy to view ordinal prediction as equivalent to a ranking problem (or so-called *multipartite ranking* but with ties allowed (Fürnkranz, Hüllermeier, and Vanderlooy, 2009)), Silva, Pinto, and Cardoso (2018) suggest that purely ranking-based evaluations or training objectives go too far in abstracting away from the labeled classes. To me, this observation highlights that "grouped continuous" labels are likely to be closer to "ratio" data that implicitly preserves meaningful distance between bins, and reminds me that "assessed" ordinal labels are more liable to be inconsistent between bins and from annotator to annotator.

Finally, a growing field of *monotonic classification* (Ben-David, Sterling, and Pao, 1989; Pedro Antonio Gutiérrez and García, 2016; Cano et al., 2019) makes clear use of assumed meaning in the label space to define monotonicity constraints that either incorporate domain knowledge to hopefully improve accuracy or that ensure particular forms of "fairness" in the output. This is a special case of (ordinal) classification with inputs (or "patters") $x \in \mathcal{X}$ and outputs $y \in \mathcal{Y}$. For monotonic classification, some subset $\mathcal{S}$ of the input dimensions, having ordered domains, imply a partial order on the inputs themselves as $x \succeq x' \Leftrightarrow x_s \succeq x'_s \forall_{s \in \mathcal{S}}$, giving rise to the following constraint on prediction $x \succeq x' \Rightarrow y \Rightarrow y'$. Clearly such constraints require ordering to be available for the label-set $\mathcal{Y}$ even if it might not always be a useful bias for prediction (c.f. (Ben-David, Sterling, and Tran, 2009)).

To summarize, the interpretation of ordinality suggests that possible ways to incorporate additional bias into your model and evaluation with the hope of reducing variance with respect to information of interest. Whether such bias is helpful or harmful will depend on particular circumstances.

### 5.4.2 Additional Examples and Approaches in Ordinal Regression

In the context of these interpretations of ordinality, consider the following sample of approaches to training ordinal regression models.

Beckham and Pal (2016) propose the least squares regression of the expectation under what has the structure of a multi-nominal logistic regression model (treating the ordinal categories as numeric so that an expectation can be computed). [8] For comparison, they also consider replacing the expectation $a^T f(x)$ where $a = [0, 1, 2, ..., k-1]$ and $f(x)$ is a normalized discrete distribution with a different arbitrary weighted sum, $a'^T f(x)$ with the vector $a'$ being learned simultaneously during the regression. This can be thought of

[8] They formulate the model as a normal distribution with fixed standard deviation and with mean given by the expectation under this linear predictor, but the normal distribution does not factor into the training nor into the decoding since they decode by rounding the expectation to the nearest integer value.

as learning to project the ordinal labels to numeric values which is a theme of a number of previous works, but has the consequence of making the optimization problem non-convex.

Jianlin Cheng, Zheng Wang, and Pollastri (2008) try to capture an ordinal-relevant loss function by coding outputs as binary prefix vectors. It doesn't actually enforce monotonicity, so they ignore discrepancies by considering labels in a fixed order.

W. Jiang (2018) extend linear discriminant analysis to view ordinal regression. Their starting point is a method that seeks a linear projection of features so as to simultaneously minimize the projected difference of each input vector from the average vector from the same category as well as the 2-norm of the weight vector used for the projection (i.e. regularization) while maximizing a global, minimum required distance between projected adjacent category mean vectors. Such a minimum constraint between adjacent categories consequently enforces a linearly growing constraint between non-adjacent categories, and the innovation of this paper is to enforce more general constraints on label pairs. In particular, they employ an exponential distance constraints (rather than $w^T m_{k+d} - w^T m_k > d\rho$, they enforce $w^T m_{k+d} - w^T m_k > exp(d)\rho$). In their experiments, the additional inductive bias led to modest gains over the baseline LDA approach which was, itself, consistently better than the other baselines compared against. Dobrska, H. Wang, and Blackburn (2012) follow similar intuitions but from the pairwise perspective. Rather than learning binary classifiers that correctly rank instances with higher ordinal labels above those with lower labels, they train a pairwise regressor to predict the distance between two instances with "distance" being defined as the square root of the euclidean distance between the centroids from the training data multiplied by the sign of the label difference (though they point to the possibility of defining distance without using label centroids). Training is expensive because of the quadratic number of pairwise distances to learn, and prediction is linear in the number of training examples. At test time, each training example has a label and, therefore, a monotonically decreasing target preference from its true label to each of the other candidate labels, so that each possible label has a corresponding range of preferences. The model's preference between the training example and the test example will correspond to one of these labels. Prediction is done by taking a majority vote after having each training instance cast a vote. Liu et al. (2011) modifies the LDA approach so that, rather than minimizing the distance of points to the mean of their label, the K-Nearest Neighbors (KNN) graph is consulted and all points that are mutual KNN neighbors are encouraged to be close to each other (dropping the need for explicit regularization of the weight vector). However, this objective decays exponentially as the distance in label space grows. Liu et al. (2012) begin with the same framework, but allow additional, orthogonal projections to be learned as well. The final decision is made via majority across all dimensions.

Rennie (2005) investigate two dimensions of signal for linear models of ordinal regression: a margin violation penalty and a construction that aggregates penalties across possible classes. The penalty functions take the distance from the linear prediction $z$ to a boundary and return a penalty. The smoothed hinge penalty is a roughly linear cost with the distance from the margin boundary and the modified least squares is roughly quadratic (but both are differentiable and become 0 when the margin constraints are satisfied). The logistic penalty in the binary case amounts to the negative log-likelihood of a logistic conditional model, but it can be thought of as a margin penalty in the sense that we receive a penalty as long as the probability of the correct class is less than one. In fact, given that logistic regression is typically regularized by imposing an penalty on the squared $\ell_2$ norm of the weight vector, the authors cast the logistic penalty as $e^{\frac{1}{2}|z-y|}$ They do not include results of the ordistic model in their experiments and it doesn't seem to have been used much since (although Sun, Nagaraj, and Westover (2018) is an exception). In any case, the all-thresholds model (which accentuates a *growing* difference between categories—a theme in a number of papers) works best in their experiments, while Fathony, Bashiri, and Ziebart (2017) actually had the opposite findings.

Fürnkranz, Hüllermeier, and Vanderlooy (2009) contrast the task of ordinal regression (as a classification problem) with multipartite ranking (the corresponding ranking problem that does not require boundaries to be explicitly known). They compare two binary decompositions of the problem: *Frank and Hall* versus *All-pairs* and find that the former is more effective for ranking and the latter for classification on the datasets they use. They highlight the use of C-index and the Jonckheere-Terpstra statistic which are related to multi-class extensions of the area under the ROC curve.

### 5.4.3 Similar Approaches

Fernandez-Gonzalez, Bielza, and Larranaga (2015) builds from the general formulation of Gaag and Waal (2006) and Benjumeda, Bielza, and Larrañaga (2016) to explore a multi-dimensional problem in which at least one of the dimensions is ordinal, but most are nominal. The focus of the framework is achieving an expressive modeling distribution that is still tractable by leveraging the fact that the features will be observed at prediction time. They use a Gaussian Bayesian Network Classifier which has some connections to the CRFs that we use. An important distinction, however, is that their models are directed, inference is exact, and subsets of features are assumed to be jointly distributed given some subset of class variables, and while our CRFs are undirected, conditional models, and I am often satisfied with approximate inference. Sutton and McCallum (2012) collect a multi-label (non-ordinal) corpus of user reactions that could have been ordinal. Similarly, Phan, Shindo, and Matsumoto (2016) curate an interesting multi-label (but still binary) dataset of emotions expressed during movie dialog using the emotional dimensions proposed by Robertf Plutchik (1980) (see also Robert Plutchik (2001)).

Twomey et al. (2019) cast ordinal regression as a linear-chain CRF of length equal to the number of ordinal levels. This appears to be a CRF formulation of the adjacent categories model. The joint probability is defined as the product of the probabilities $P(Y >= j|Y >= j-1) \propto \exp(XW_j)$. The gradient calculation seems to confirm that the model is equivalent to the nominal representation. Note that, while they employ a CRF, the structure of their model is to achieve a single ordinal prediction.

While most work in ordinal regression has focused on single-output regression, the notable work of Kim and Pavlovic (2010) builds a structured model—indeed a CRF—where they simply substitute out the standard log-linear unary potential functions with a log-*non-linear* function that represents the probability of each ordinal label given a mean that is linear in the observed features plus Gaussian noise with learned standard deviation and learned bin boundaries. Importantly, they leverage the ordinal nature of variables to improve representation at the unary factors, but apparently in this work make no modification to standard higher-order factors (although it is possible with effort to extend the same principle to work with higher-order factors).

Several papers suggest that inconsistencies present in the binary outputs used by ordinal reduction methods are a shortcoming of those methods (e.g. (Cao, Mirjalili, and Raschka, 2019)), though they do not explain why. My perspective is that such inconsistencies merely represent a non-linear decision boundary that may very-well be more robust than alternatives. However, it may be worth considering alternative aggregation methods like a soft-min, or harmonic mean.

While most ordinal work does not deal with structured prediction, Cheng, Dembczynski, and Hüllermeier (2010) introduce the term “graded, multi-label”, and propose two flavors of meta techniques for reducing graded multi-label classification either to binary multi-label or to ordinal single-label (or a hybrid). The experimental analysis specifically asks about the usefulness of graded signal including for the case where binary labels are ultimately desired (c.f. Chapter 7 in this thesis).

## 5.5 Conclusion

I have presented ordinal and partially ordinal decompositional models of semantics and have shown how they can use a CRF framework to fit data. In the future, I would like to investigate the propensity of our models to better enforce (and admit strict enforcement) of structural consistency constraints. For example, the universal decompositional semantics effort has labeled predicates with respect to factuality—how likely is it that the author believed that the event represented by the predicate actually occurred; physicality—how likely is it that the identified argument was a physical entity; physical contact—how likely is it that the object made physical contact during the course of the event described by the predicate. We could further annotate argument-argument pairs with respect to whether they made physical contact with each other, whether one caused a change in the state of the other, etc. Similarly, we could label emotional contact, emotional change of state, and emotional properties with respect to before and after the event.

I would expect different patterns of errors if we predict such inter-related properties independently than if we were to predict them jointly. While we would hope that a neural model would pick up on such inter-relations, in the graphical models framework, we can explicitly enforce or flexibly model specific interactions of interest.

# Chapter 6

# BP Inference with TorchFactors

Focusing on only a subset of the SPR properties enabled tractable comparisons of various models with exact inference, and the results showed the importance of the jointly connected models even on that small subset and showed at least modest gains from the loopy model over the carefully selected tree-based model. For models of more variables, brute-force inference quickly becomes intractable as does exact inference on general loopy models. Loopy belief propagation has been an important method for approximate inference in undirected graphical models and generalizations have been discovered which allow even more trade-offs between runtime complexity and approximation quality. In addition to providing a means for predicting most likely labels under a model, belief propagation offers an approximation of the partition function for a particular input which can then be used to approximate the likelihood of the data under the model. Loopy belief propagation, however, is not guaranteed to converge to a fixed-point, and in prior preliminary experiments using loopy belief propagation, it appeared to me that following the gradient of the loopy BP log-likelihood approximation results in surprising, degenerate behavior where the approximate log-likelihood can be maximize by hindering the quality of the convergence and approximation rather than by improving the true likelihood of the data under the model. Nevertheless, the same algorithm applied to tree-structured models admits exact inference and exact computation of the loglikelihood.

This chapter introduces the belief-propagation-based exact and approximate inference capabilities of the tx library and then investigates the above degeneracy issue within the context of the SPRL task. First, I compare the performance of BP-based *inference* using models that were trained with exact inference on the same subset of the problem used in previous chapters. Next, since BP can be used for efficient exact inference in tree-structured models, I train such a model using BP based on a maximum-likelihood bias tree as was done in Chapter 4. Finally, I show the impact of using approximate inference during training and consider a non-convergence penalty as an attempt to counteract the degenerate learning.

## 6.1 Generalized BP on Cluster Graphs With TorchFactors

The torchfactors library implements differentiable belief propagation on cluster graphs. In cluster graphs, there is only a single type of node—a *cluster* containing any subset of variables and factors. The homogeneous nature of this definition makes the definition of message passing particularly simple. At any time, there is a current message associated with each directed edge in the the undirected cluster graph. A message takes the same form as a factor. Sending a message from $C_i$ to $C_j$ simply amounts to multiplying all factors in cluster $C_i$ with all incoming messages from other clusters and then marginalizing across all be the variables shared by both $C_i$ and $C_j$. As discussed in Chapter 5, tx factors need only know how to perform sum-product operations over local information (in log space). For the dense factors used in this thesis, the crux of this operation is to create a two-dimensional view of each tensor with columns corresponding to configurations of the variables that are to be summed over and rows corresponding to configurations of the variables to be retained. These matrix layers are stacked into a single 3-dimensional which is then (log) multiplied element-wise across layers to form a matrix which is (log) summed across columns resulting in a vector which is then viewed again as a k-dimensional output tensor. Because of this formulation, the entire message passing procedure is automatically differentiable using pytorch.

Although the framework allows for more general cluster graphs, my experiments use the beta cluster graph—one cluster for each variable and one cluster for each factor— which gives us belief propagation as it would be on the factor graph. To avoid unnecessary cycles in the cluster graph, I greedily merge clusters with a neighbor if the neighbor's scope is a superset of its scope. The junction tree algorithm can be used to form a cluster-graph that is tree-structured despite loops in the underlying model allowing exact inference via BP that scales exponentially in the tree-width of the loopy factor-graph. Yedidia, Freeman, and Y. Weiss (2005) showed how BP message passing found stable local optima of an approximation to the partition function and gave a technique for constructing generalized message passing algorithms to optimizing increasingly accurate approximations. tx was designed to support a wide generality of message passing approaches that includes both the cluster-graph generalization as well as the generalized BP family of region-based approximations.

Thus, following a body of landmark work (e.g. (Stoyanov, Ropson, and Eisner, 2011; Domke, 2013)), inference via belief propagation simply becomes a differentiable, parameterized black box module to apply to an input representation. In such an automatically differentiated framework, we can easily employ approximations such as skipping messages that represent only a small delta (c.f. Domke (2011) who learn using a truncated number of message passing iterations) while still computing a true (stochastic) gradient of our objective function. [1]

## 6.2 Exact vs Approximate Inference—Test Only

Both as a way of isolating any degenerate learning from degraded performance from approximation, I first consider evaluating the final models from Chapter 4 using belief propagation as the inferences at test time. For tree-structured models, an appropriate choice of message-passing schedule can achieve exact marginals after only sending each message once. For simplicity of implementation, I used a less-optimal message-passing schedule that converges on trees after two passes. For SPRL⋆ and SPRL⋆0, marginals are not generally guaranteed to converge, so I evaluate performance after two and five passes of belief propagation.

Tables 6.1 and 6.2 compare, for validation and dev data respectively, the results of

[1] A potentially interesting by-product of a differentiable message-passing implementation could be the ability to monitor the relative importance of the messages with respect to the computed loss. For loopy cluster-graphs, belief propagation may require several iterations to achieve convergence, and, indeed, it may never converge. The message-passing *schedule* may have a large impact on convergence and investigating the gradients may prove fruitful in approximate inference via message passing.

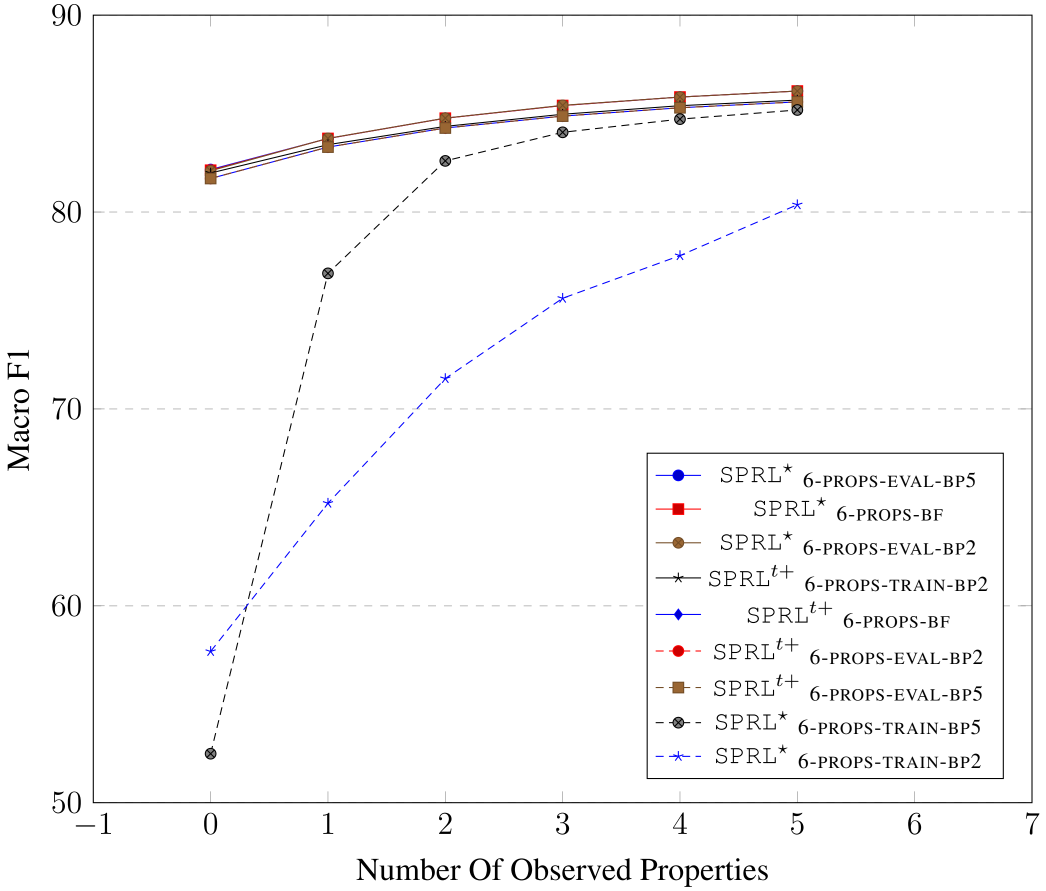


**Figure 6.1.** Conditional macro f1 on val data using BP inference for some models (6 props)

evaluating the various models with brute force against the results using belief propagation at test time. Essentially there is no loss of performance when using approximate inference at test time. Section 6.3 discusses the rows with "train" in the subscript.

Figures 6.1 and 6.2 show similar results on the conditional prediction task (Figures 6.3 and 6.4 show the same but excluding the lowest performing scenarios for better resolution). Section 6.3 discusses the curves with "train" in the subscript.

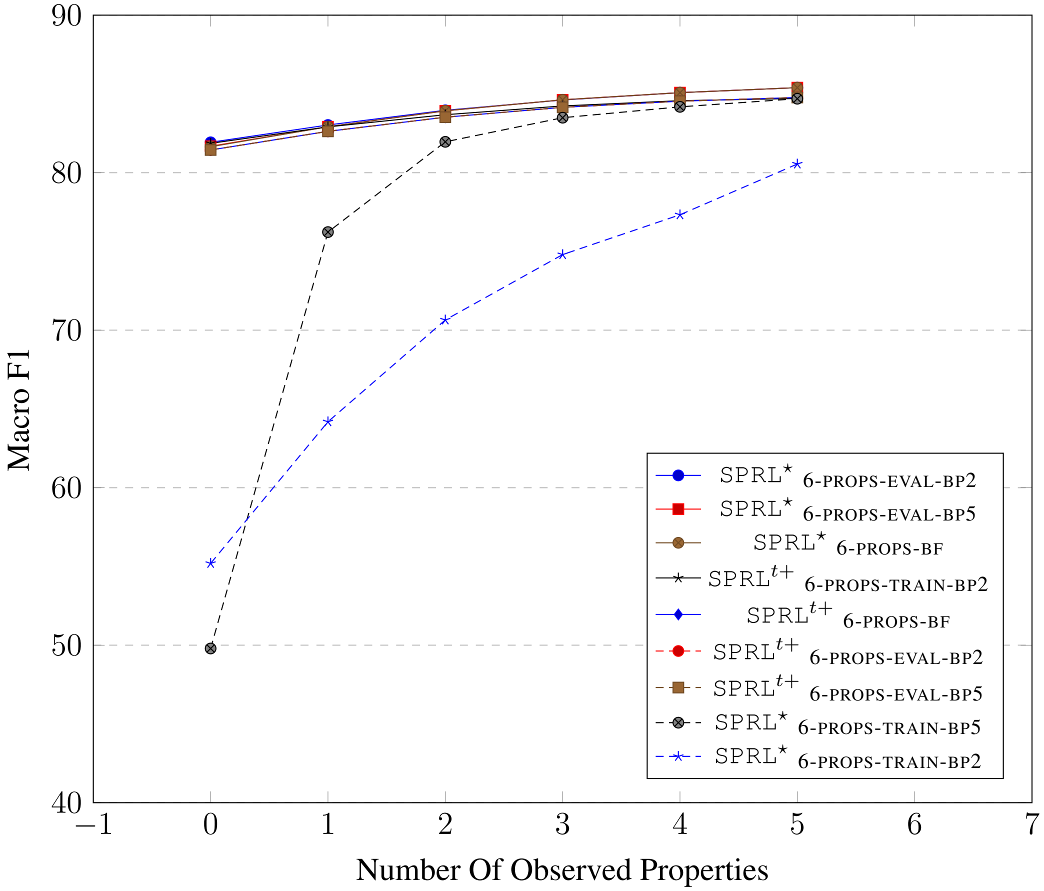


**Figure 6.2.** Conditional macro f1 on dev data using BP inference for some models (6 props)

## 6.3 Approximate Inference—Training on Subset SPRL

Since BP computes exact marginals and likelihood for tree-based models, training such models with BP inference achieves identical results to training with brute-force inference. This can be seen by the rows and curves with SPRL$^{t+}$ sub-scripted with “train-BP-2”. However, rows and curves with results for SPRL$^\star$ sub-scripted with “train-BP” show that the model trained under approximate inference is actually much worse than the model trained under exact inference but evaluated using the approximation. Recall that early stopping and

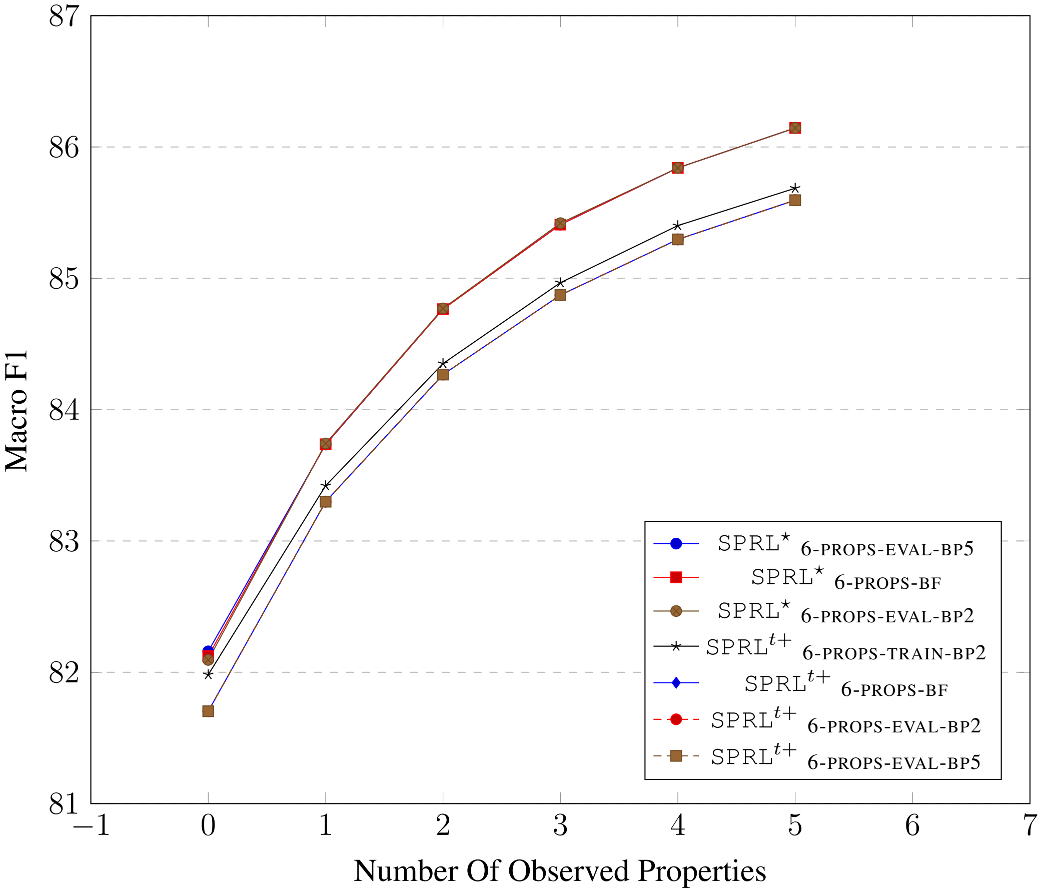


**Figure 6.3.** Conditional macro f1 on val data using BP inference for some models (top models only – 6 props)

hyper-parameter optimization is all driven by model likelihood which becomes a move-able target. The good performance of loopy BP using the parameters obtained from the exact model tells us that the issue is a limitation of the learning rather than an inherent limitation of the inference method.

The 2-pass BP model has especially erratic behavior. The extreme likelihood is probably a symptom that the model is learning to sabotage quick convergence in order to allow it to over-estimate the true likelihood and thereby please the training objective. It is also interesting that the micro and macro F1 scores of this model (on both validation and development data) are uncommonly different from one another. The 5-pass model, on the

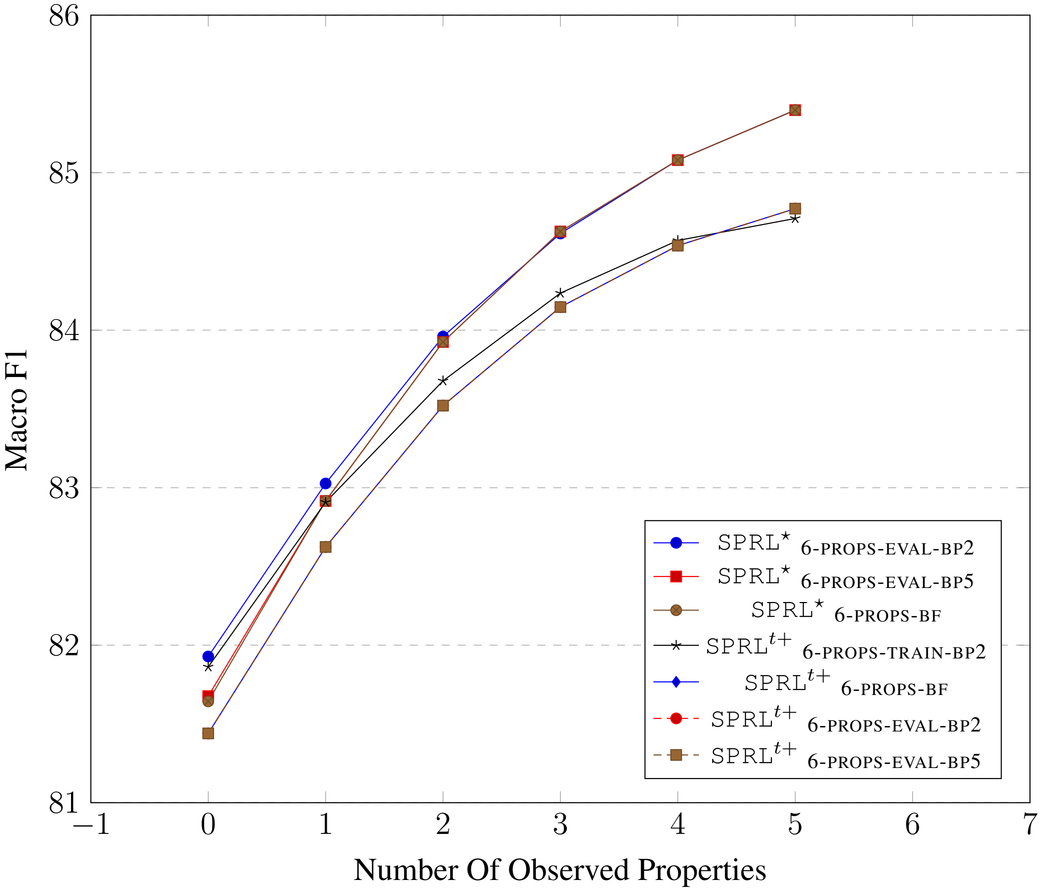


**Figure 6.4.** Conditional macro f1 on dev data using BP inference for some models (top models only – 6 props)

other hand, has a more reasonable (although certainly inflated) likelihood, more uniform micro and macro f1, and conditional plots showing that it makes good use of observed labels (though not enough to surpass the BP-trained tree-based model. As expected, the more passes of inference allow more information to flow between properties and force the model to achieve a level of convergence that helps to stabilize the likelihood estimate better.

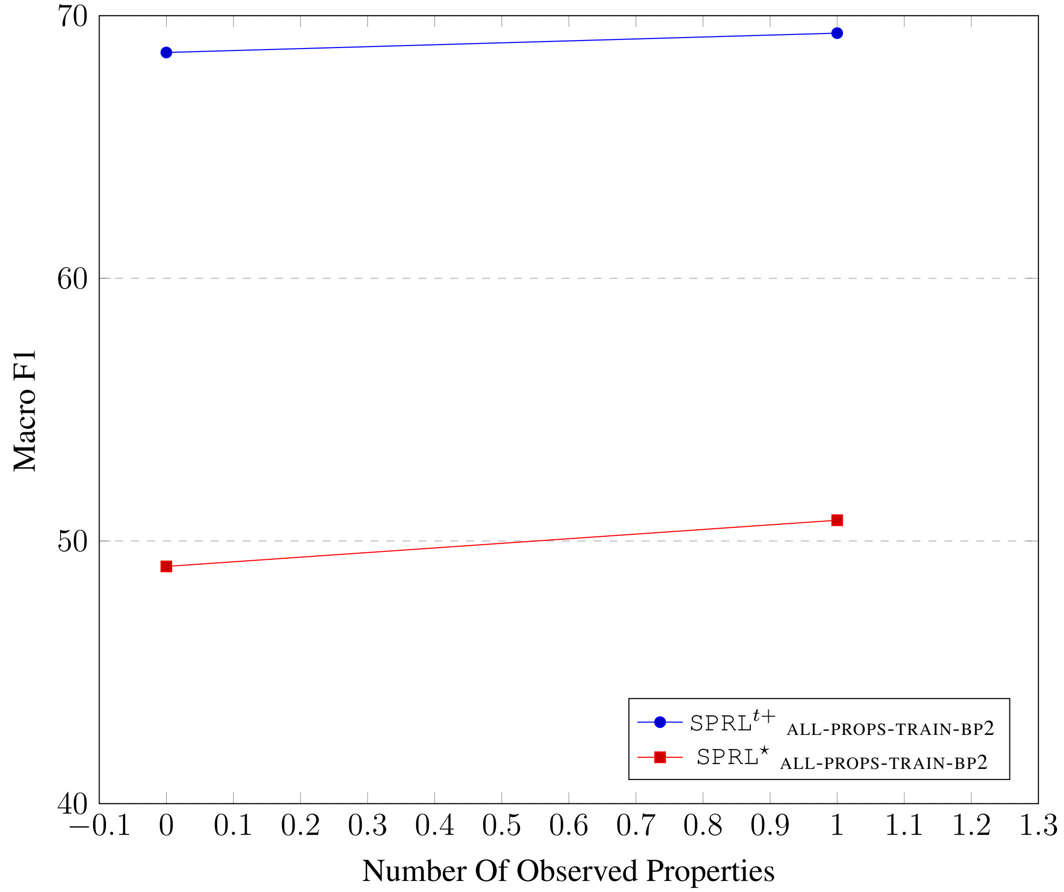


**Figure 6.5.** Conditional macro f1 on val data using BP inference for some models (all props)

## 6.4 Full SPRL

Tables 6.3 and 6.4 and Figures 6.5 through 6.10 show results on the full SPRL task from Chapter 3 as well as a partial evaluation of conditional inference under the SPRL$^{t+}$ model. As with the six-property subset of the problem, the loopy-BP based training obscures the true likelihood objective. The tree-based model is able to improve significantly over the independent model with respect to likelihood but only minor improvements in terms of F1; however, it is able to make use of test-time label observations while the independent model cannot. If the degenerate training of the loopy-model were overcome, I expect that it would make even better use of conditionally observed test-time labels.

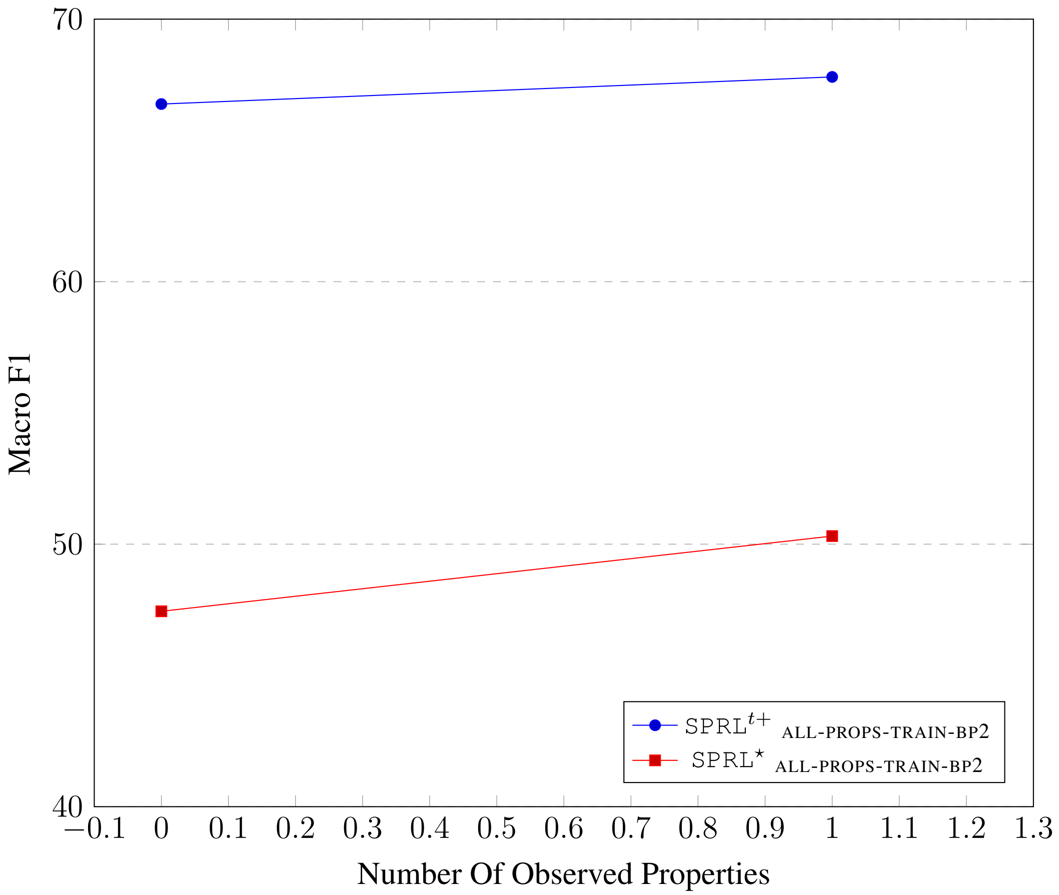


**Figure 6.6.** Conditional macro f1 on dev data using BP inference for some models (all props)

## 6.5 Non-Convergence Penalty

I leave in-depth investigation of this phenomenon for future work, but conclude this chapter with a proposal for combating this degenerate behavior in learning. Given my hypothesis that non-convergence of the model is responsible for the poor likelihood estimates, I impose a penalty on pending messages. If messages ever reach a fixed point (as they do for tree-based models), the pending message and the previously sent message are identical. The more recent message is presumably a better model of the information that should be carried over that channel so a possible penalty can be imposed based on the KL divergence

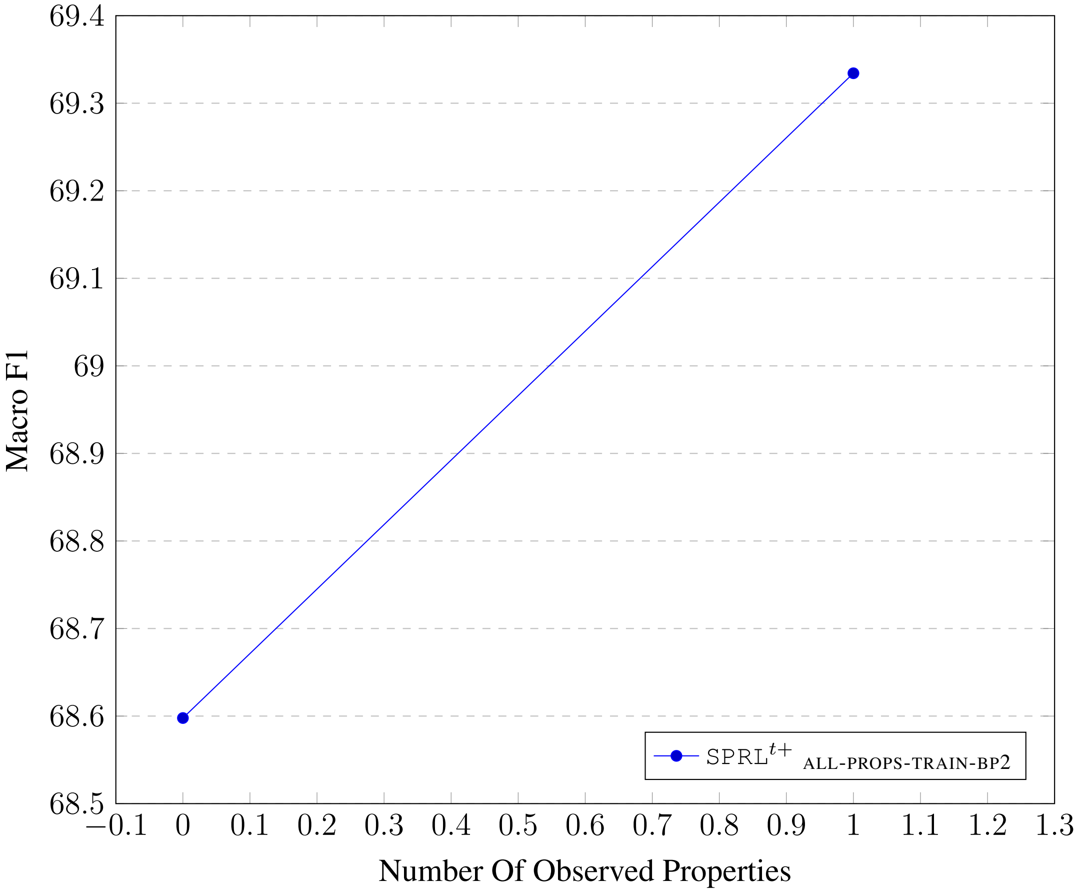


**Figure 6.7.** Conditional macro f1 on val data using BP inference for some models (top models only – all props)

between the most recent message and the previously sent message.[2] The KL will be 0 for a system that has had no change in the previous pass of messages. Care must be taken in ensuring that the penalty is incorporated in a way that doesn't dominate the objective since convergence with a fixed number of BP iterations may be impossible and too strong of a penalty will encourage an uninformative distribution that needs no message passing to achieve convergence.

As an example, for a hyperparameter $c$ and where $m'_{st}$ is the most recent message sent from region $s$ to region $t$ and $m''_{st}$ is the message sent before that, the following term could

[2] A more accurate penalty would be to compute the KL between the pending message and the most recently sent, but that requires computing a round of messages that does not get passed.

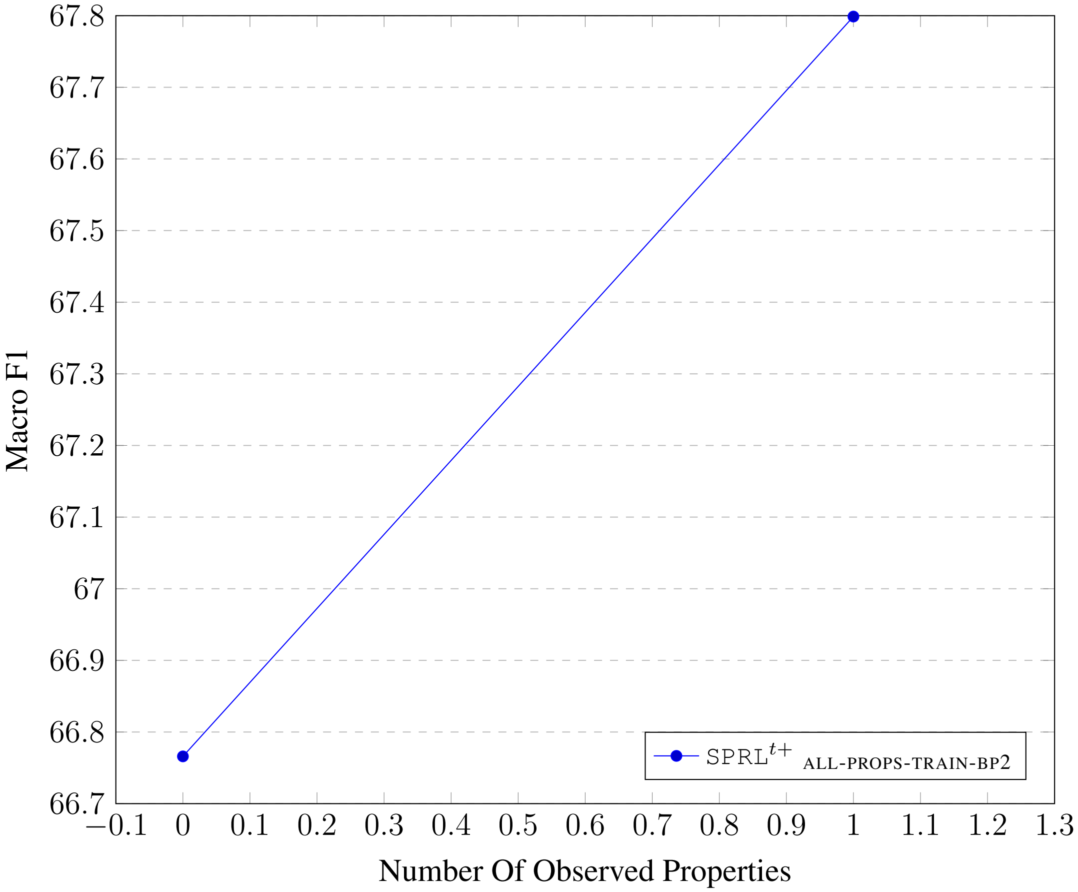


**Figure 6.8.** Conditional macro f1 on dev data using BP inference for some models (top models only – all props)

be added to the negative log-likelihood training objective to encourage convergence:[3]

$$c \log \sum_{st} \exp KL(m'_{st} \| m''_{st})$$

## 6.6 Conclusion

This chapter has introduced approximate inference capabilities of my †x library for the purpose of scaling up joint models of SPRL. It has evaluated the impact of approximate

[3] I leave the evaluation of this penalty term and the question of how to choose the constant $c$ for future work.

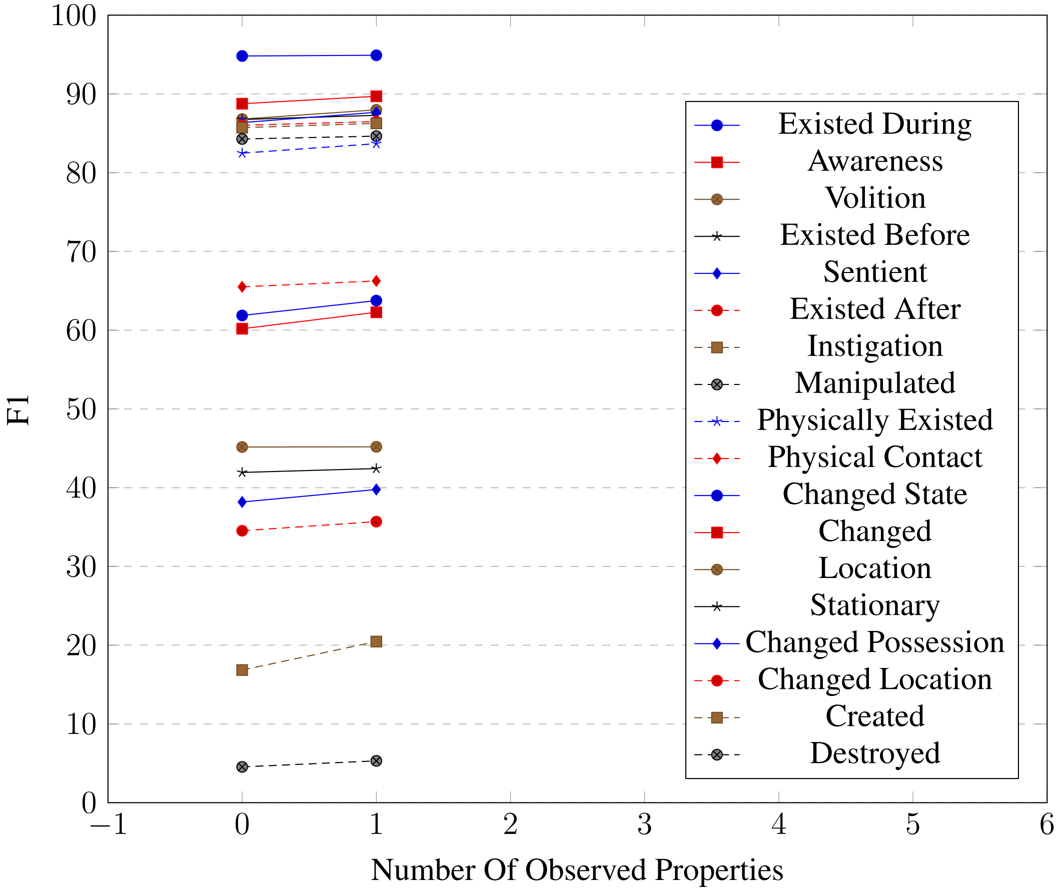


**Figure 6.9.** Property-specific conditional f1 for val data using SPRL$^{t+}$ ALL-PROPS-TRAIN-BP2

inference using loopy models trained with exact inference. It gave results from the SPRL$^{t+}$ model for the full SPRL task trained in †x via BP and highlighted the degenerate nature of training based on non-converged belief propagation using its approximate loglikelihood as the objective. Future work should investigate whether the gap between the tree-based and exact-trained pairwise model widens or narrows as the number of properties in the joint model grows.

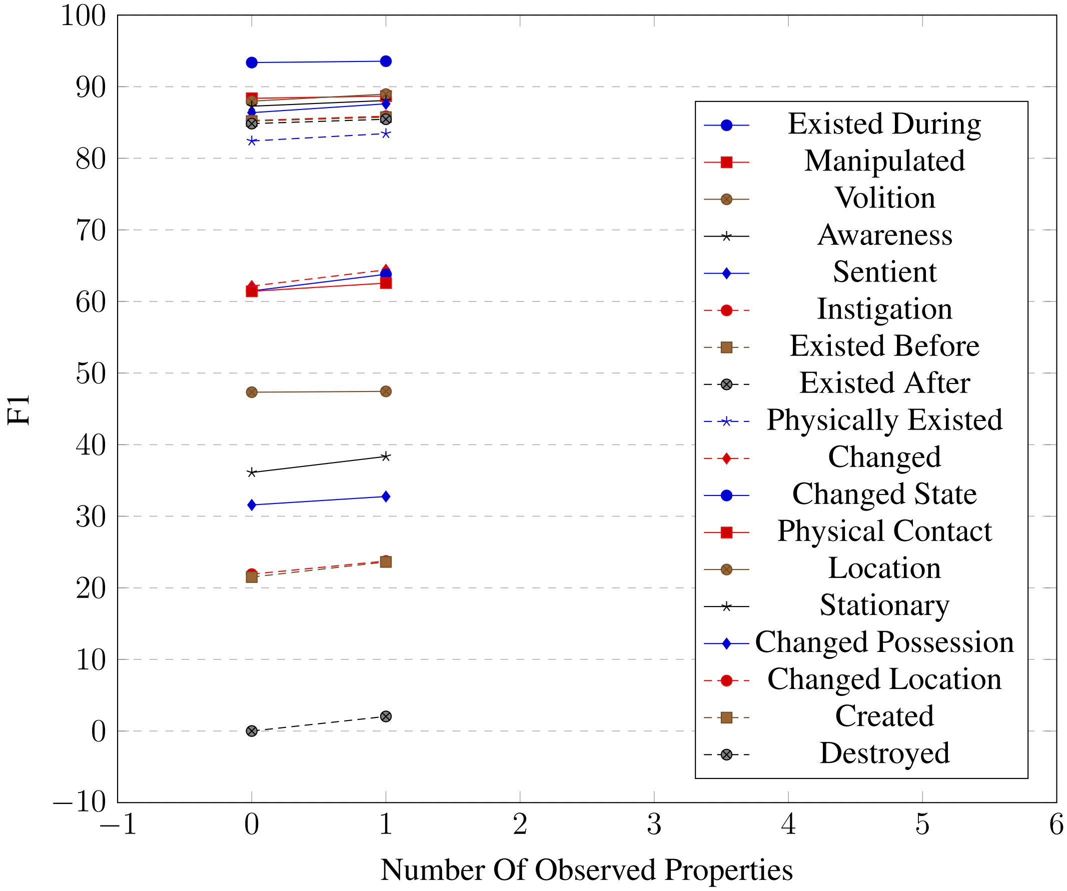


**Figure 6.10.** Property-specific conditional f1 for dev data using SPRL$^{t+}$ ALL-PROPS-TRAIN-BP2

| | | | **averaging=micro** | | | **averaging=macro** | | |
|---|---|---|---|---|---|---|---|---|
| | **setting** | **likelihood** | F1 | P | R | F1 | P | R |
| 0 | SPRL$^{\star}$0 6-PROPS-BF | 6.2 | 39.9 | 70.2 | 27.8 | 28.4 | 95.0 | 16.7 |
| 1 | SPRL$^{\star}$0 6-PROPS-EVAL-BP2 | 7.5 | 39.9 | 70.2 | 27.8 | 28.4 | 95.0 | 16.7 |
| 2 | SPRL$^{\star}$0 6-PROPS-EVAL-BP5 | 7.5 | 39.9 | 70.2 | 27.8 | 28.4 | 95.0 | 16.7 |
| 3 | SPRL 6-PROPS-BF | 14.3 | 82.5 | 82.8 | 82.1 | 81.9 | 82.7 | 81.1 |
| 4 | SPRL 6-PROPS-EVAL-BP2 | 14.3 | 82.5 | 82.8 | 82.1 | 81.9 | 82.7 | 81.1 |
| 5 | SPRL 6-PROPS-EVAL-BP5 | 14.3 | 82.5 | 82.8 | 82.1 | 81.9 | 82.7 | 81.1 |
| 6 | SPRL 6-PROPS-TRAIN-BP2 | 14.2 | 82.4 | 82.8 | 82.0 | 81.7 | 82.6 | 80.8 |
| 7 | SPRL$^{t-}$ 6-PROPS-BF | 14.1 | 82.6 | 83.0 | 82.2 | 82.1 | 83.1 | 81.1 |
| 8 | SPRL$^{t-}$ 6-PROPS-EVAL-BP2 | 14.1 | 82.6 | 83.0 | 82.2 | 82.1 | 83.1 | 81.1 |
| 9 | SPRL$^{t-}$ 6-PROPS-EVAL-BP5 | 14.1 | 82.6 | 83.0 | 82.2 | 82.1 | 83.1 | 81.1 |
| 10 | SPRL$^{t+}$ 6-PROPS-BF | 18.2 | 82.4 | 83.2 | 81.7 | 81.7 | 83.0 | 80.5 |
| 11 | SPRL$^{t+}$ 6-PROPS-EVAL-BP2 | 18.2 | 82.4 | 83.2 | 81.7 | 81.7 | 83.0 | 80.5 |
| 12 | SPRL$^{t+}$ 6-PROPS-EVAL-BP5 | 18.2 | 82.4 | 83.2 | 81.7 | 81.7 | 83.0 | 80.5 |
| 13 | SPRL$^{t+}$ 6-PROPS-TRAIN-BP2 | 18.4 | 82.7 | 83.3 | 82.1 | 82.0 | 83.1 | 80.9 |
| 14 | SPRL$^{\star}$ 6-PROPS-BF | 18.9 | 82.8 | 83.6 | 82.0 | 82.1 | 83.5 | 80.8 |
| 15 | SPRL$^{\star}$ 6-PROPS-EVAL-BP2 | 19.7 | 82.7 | 83.5 | 82.0 | 82.1 | 83.4 | 80.8 |
| 16 | SPRL$^{\star}$ 6-PROPS-EVAL-BP5 | 19.4 | 82.8 | 83.6 | 82.0 | 82.2 | 83.5 | 80.9 |
| 17 | SPRL$^{\star}$ 6-PROPS-TRAIN-BP2 | 98.6 | 44.7 | 42.6 | 46.9 | 57.7 | 61.3 | 54.5 |
| 18 | SPRL$^{\star}$ 6-PROPS-TRAIN-BP5 | 24.4 | 54.1 | 75.9 | 42.1 | 52.5 | 78.3 | 39.5 |

**Table 6.1.** System and inference comparison on val data (6 props)

| | | | averaging=micro | | | averaging=macro | | |
|---|---|---|---|---|---|---|---|---|
| | **setting** | **likelihood** | F1 | P | R | F1 | P | R |
| 0 | SPRL$^{\star}$0 6-PROPS-BF | 6.3 | 39.7 | 68.1 | 28.0 | 28.3 | 94.7 | 16.7 |
| 1 | SPRL$^{\star}$0 6-PROPS-EVAL-BP2 | 7.6 | 39.7 | 68.1 | 28.0 | 28.3 | 94.7 | 16.7 |
| 2 | SPRL$^{\star}$0 6-PROPS-EVAL-BP5 | 7.6 | 39.7 | 68.1 | 28.0 | 28.3 | 94.7 | 16.7 |
| 3 | SPRL 6-PROPS-BF | 14.0 | 81.7 | 82.1 | 81.3 | 81.3 | 82.1 | 80.6 |
| 4 | SPRL 6-PROPS-EVAL-BP2 | 14.0 | 81.7 | 82.1 | 81.3 | 81.3 | 82.1 | 80.6 |
| 5 | SPRL 6-PROPS-EVAL-BP5 | 14.0 | 81.7 | 82.1 | 81.3 | 81.3 | 82.1 | 80.6 |
| 6 | SPRL 6-PROPS-TRAIN-BP2 | 14.0 | 82.1 | 82.5 | 81.6 | 81.7 | 82.5 | 80.9 |
| 7 | SPRL$^{t-}$ 6-PROPS-BF | 14.0 | 81.6 | 82.2 | 81.1 | 81.3 | 82.3 | 80.2 |
| 8 | SPRL$^{t-}$ 6-PROPS-EVAL-BP2 | 14.0 | 81.6 | 82.2 | 81.1 | 81.3 | 82.3 | 80.2 |
| 9 | SPRL$^{t-}$ 6-PROPS-EVAL-BP5 | 14.0 | 81.6 | 82.2 | 81.1 | 81.3 | 82.3 | 80.2 |
| 10 | SPRL$^{t+}$ 6-PROPS-BF | 18.2 | 82.0 | 83.8 | 80.3 | 81.4 | 83.7 | 79.3 |
| 11 | SPRL$^{t+}$ 6-PROPS-EVAL-BP2 | 18.2 | 82.0 | 83.8 | 80.3 | 81.4 | 83.7 | 79.3 |
| 12 | SPRL$^{t+}$ 6-PROPS-EVAL-BP5 | 18.2 | 82.0 | 83.8 | 80.3 | 81.4 | 83.7 | 79.3 |
| 13 | SPRL$^{t+}$ 6-PROPS-TRAIN-BP2 | 18.3 | 82.4 | 83.8 | 81.0 | 81.9 | 83.7 | 80.1 |
| 14 | SPRL$^{\star}$ 6-PROPS-BF | 18.9 | 82.1 | 83.4 | 80.7 | 81.6 | 83.5 | 79.9 |
| 15 | SPRL$^{\star}$ 6-PROPS-EVAL-BP2 | 19.6 | 82.3 | 83.6 | 81.0 | 81.9 | 83.8 | 80.2 |
| 16 | SPRL$^{\star}$ 6-PROPS-EVAL-BP5 | 19.3 | 82.1 | 83.4 | 80.8 | 81.7 | 83.5 | 79.9 |
| 17 | SPRL$^{\star}$ 6-PROPS-TRAIN-BP2 | 98.1 | 43.1 | 40.7 | 45.9 | 55.2 | 57.1 | 53.4 |
| 18 | SPRL$^{\star}$ 6-PROPS-TRAIN-BP5 | 24.3 | 51.3 | 73.8 | 39.3 | 49.8 | 77.2 | 36.7 |

**Table 6.2.** System and inference comparison on dev data (6 props)

| | | | **averaging=micro** | | | **averaging=macro** | | |
|---|---|---|---|---|---|---|---|---|
| | **setting** | **likelihood** | F1 | P | R | F1 | P | R |
| 0 | $\mathtt{SPRL}_{\text{ALL-PROPS-TRAIN-BP2}}$ | 0.8 | 81.4 | 82.9 | 79.8 | 68.2 | 78.4 | 60.3 |
| 1 | $\mathtt{SPRL}^{t+}_{\text{ALL-PROPS-TRAIN-BP2}}$ | 2.4 | 81.6 | 83.3 | 79.9 | 68.6 | 79.8 | 60.2 |
| 2 | $\mathtt{SPRL}^{\star}_{\text{ALL-PROPS-TRAIN-BP2}}$ | 99.2 | 31.1 | 27.4 | 36.1 | 49.0 | 46.2 | 52.3 |

**Table 6.3.** System and inference comparison on val data (all props)

| | | | **averaging=micro** | | | **averaging=macro** | | |
|---|---|---|---|---|---|---|---|---|
| | **setting** | **likelihood** | F1 | P | R | F1 | P | R |
| 0 | $\mathtt{SPRL}_{\text{ALL-PROPS-TRAIN-BP2}}$ | 0.7 | 80.9 | 82.9 | 79.0 | 66.1 | 75.6 | 58.8 |
| 1 | $\mathtt{SPRL}^{t+}_{\text{ALL-PROPS-TRAIN-BP2}}$ | 2.5 | 81.2 | 83.5 | 78.9 | 66.8 | 77.0 | 58.9 |
| 2 | $\mathtt{SPRL}^{\star}_{\text{ALL-PROPS-TRAIN-BP2}}$ | 99.2 | 30.9 | 27.1 | 36.0 | 47.4 | 43.9 | 51.6 |

**Table 6.4.** System and inference comparison on dev data (all props)

# Chapter 7

# Notes on Data and Evaluation

## 7.1 Introduction

In Chapter 1, I mentioned a variety of components for a predictive system:

a) Data

b) Models

c) Inference

d) Optimization

e) Evaluation

Despite my primary focus on *models* in this thesis (e.g. Chapters 3 and 5), unexpected results can arise from failings of any of the above components. Chapters 4 and 6 investigated the significance of variations of inference and learning. This chapter briefly touches on issues that have arisen with respect to the final two related components: *data* and *evaluation*.

Thus far, my experiments have dealt with models of the crowd-sourced annotations from Reisinger et al. (2015) where a single, trusted annotator provided assessed ordinal ratings. But how can we evaluate the quality of the data? What is human performance on the annotation task, and how much variation would we expect in subsequently repeated annotation efforts using the same protocol? Would other annotation protocols improve human performance, lower variance, and increase annotation speed?

Without attempting thorough experiments, this chapter makes the following contributions:

1. I briefly underscore ideas from the literature regarding annotation quality.
2. I contrast the idea of *apriori* label aggregation across multiple annotators to *aposteriori* label aggregation.
3. I propose a basic method of aposteriori label aggregation within the types of CRF models used in this thesis—highlighting the case where there is a particular *target annotator*.
4. I review EASL (Sakaguchi and Van Durme, 2018) as a method of apriori label aggregation, propose a style of active learning within EASL, and consider an approach for EASL *match optimization*.

*Apriori* aggregation computes a single consensus label while *aposteriori* aggregation performs joint inference about random variables from each annotator which are finally aggregated afterward.

## 7.2 Crowd Data and Evaluation

In this section, I briefly review literature and relevant issues related to collecting annotations from crowd workers and evaluating aspects of the quality of annotations.

Relevant work in so-called “Crowd-Truth” has acknowledged the richness of the signal available from non-expert workers when those annotations disagree (c.f. Dumitrache, Inel, Aroyo, et al. (2018), Aroyo and Welty (2015), and Dumitrache, Inel, Timmermans, et al. (2018)).

However, multiple annotators can be a double-edged sword. If I am hoping to arrive at labels that approximate the average human on a task, then probing multiple human annotators may be helpful. Furthermore the ability to compare annotations across humans can reveal insights about the difficulty of the task and the degree to which we can be confident in the labels we have acquired. On the other hand, discrepancies in interpretation of the task and use of available labels can be confounding. In many cases, the goal may not be to approximate the assessment of a lay person but rather to please a small set of well informed stake holders who will ultimately set the objective.

Feyisetan et al. (2018) investigated the behavior and performance of crowd workers on the task of Named Entity Recognition (NER) which involved identifying mentions of people, organizations, and locations in tweets. A key feature of their crowd tasks was the ability to skip particular assignments. Among the aspects they study, they measured the impact of extended instructions on the accuracy of workers as well as its impact on the workers’ decision to not skip inputs. Interestingly, they found that the instructions did not improve accuracy but did improve speed and increase the likelihood that a worker would accept a task.[1] They claim to follow Bhowmick, Basu, and Mitra (2008) for agreement evaluation.

Given an annotation effort, we want to know how confident we should be that repeating the process (possibly with other annotators from the same basic population) will lead to similar labels and we would like to have an idea of human performance as a sort of target for our computational systems. Reliability and agreement are typically considered related but distinct ideas (Kottner and Streiner, 2011; Kottner et al., 2011; Vet et al., 2006). There have been a few works especially directed at agreement in natural language processing (Carletta et al., 1997; Artstein and Poesio, 2008; Antoine, Villaneau, and Lefeuvre, 2014). Some of the most popular agreement measures include Cohen's Kappa, ICC, Cronbach's alpha, and Krippendorff's alpha for nominal, binary, ordinal (interval, ratio) (Krippendorff, 2019; Krippendorff, 2011). Examples of agreement analyses can be found in Canales et al. (2016) and Raulamo-Jurvanen, Hosio, and Mäntylä (2019). Care also needs to be taken when ties are present (which is extremely common with ordinal data).

Gwet (2016) describes a method for testing significant differences in correlated agreement coefficients (e.g. from before to after a training) and summarizes several common agreement coefficients. Klein (2018) compares several agreement measures and is favorable to Gwet's AC. Hoek and Scholman (2017) investigate the appropriateness of Gwet's $AC_1$ for discourse annotation, and recommend using it alongside Cohen's kappa, discussing the anomalous behavior that happens for both under certain circumstances. One of the primary distinctions for $AC_1$ is that it accounts for the number of categories. The authors

[1] It would be interesting to better understand if the non-improvement of accuracy was a reflection of the increased difficulty of the annotating task that annotators were willing to accept given more instruction. It seems likely that annotators will be hesitant to offer annotations if they perceive a large entropy in the distribution over reasonable interpretations of the guidelines relative to the entropy in the distribution over gold labels.

further note interest in using the distribution of responses from multiple coders to help distinguish between biases, errors, or legitimate ambiguity in the data. They highlight work by Passonneau and Carpenter (2014) and Hovy et al. (2013) that suggests modeling the annotation process in ways similar to the multi-annotator models considered below.

Krippendorff (2019) gives an insightful overview of reliability in empirical scientific studies. He identifies a tension between *reliability* and *validity*, reminiscent of the famous trade-off between variance reduction and bias reduction respectively. Reliability exists within the context of a particular experiment and reflects consistency of results, while validity deals with the quality of conclusions drawn from the results. He distinguishes three types of reliability: stability—will the same humans produce the same responses later, replicability—will other humans produce the same responses, accuracy—will humans match accepted standard responses, and surrogacy—can the responses of one human or an automated algorithm be a suitable replacement for standard responses or the responses of a group. More broadly, these represent the insensitivity of experimental result to tangential details. He cites Lorr and McNair (1966) who identify a scientific problem with the evaluation of reliability after training workers and modifying instructions. Agreement studies based on new worker pools are more appropriate. Krippendorff suggests starting with the coder pair with maximum agreement $\alpha$; if sufficiently high, add the coder who reduces $\alpha$ the least until $\alpha$ falls below what is acceptable. He mentions Zillman's (1964) scale (Zillmann, 1964) from absent to "very much present".

Krippendorff's $\alpha$ offers an extremely general conceptualization for measuring annotation reliability—namely, the relative reduction in average annotation discrepancy with other annotations on the same item versus the average discrepancy with all other annotations (on any item). The idea is that reliability should measure the degree to which annotations consistently reflect item-specific information, so a perfect $\alpha = 1$ is achieved if there are no annotation discrepancies and $\alpha = 0$ if the average *intra-item* discrepancies match the average discrepancies among all annotations (ignoring which items they belong to). Since it is a measure of consistency of observation, it explicitly ignores observations that do not have an alternative observation (i.e. labels on items that were not redundantly labeled). Confidence intervals can be easily calculated by bootstrapping: repeatedly sample $n'$ items (uniformly, with replacement) from the set of all items and compute $\alpha$ on each of the resulting sets. Krippendorff (2011) clarifies the computation of $\alpha$ and suggests discrepancy functions to be used for different types of labels (nominal, ordinal, interval, etc.). Since I found the formulation hard to transparently justify, here is an alternative, though equivalent, form that I find easy to justify (assume that all items without redundant annotation are already excluded from the dataset):

$$O_{cc'} = \sum_{i=1}^{n} \frac{N_{ic}\left(N_{ic'} - \mathbb{1}\left[c = c'\right]\right)}{N_{i.} - 1} \tag{7.1}$$

$$E_{cc'} = \frac{N_{.c}\left(N_{.c'} - \mathbb{1}\left[c = c'\right]\right)}{N_{..} - 1} \tag{7.2}$$

$$\alpha_K(N, \Delta) = 1 - \frac{\langle O, \Delta \rangle_F}{\langle E, \Delta \rangle_F} \tag{7.3}$$

In short, $\{O_{cc'}\}$ represents an observed “coincidence” matrix that show the frequency with which ordered pairs of labels would be sampled from the set of labels on the same item, which $\{E_{cc'}\}$ gives the “expected coincidences”—the frequency with which ordered pairs of labels would be sampled from the entire pool of annotations (ignoring which items the labels had been assigned to). In both cases, the counts are scaled so that the total count in $O$ and in $E$ are each equal to the original total number of observations collected. The observed and expected coincidence matrices can be computed from a matrix $\{N_{ic}\}$ that gives the number of times item $i$ was assigned category $c$.[2] For notational convenience, $N_{.c}$ represents the number of times label $c$ was used on *any* item, $N_{i.}$ represents the number of times item $i$ was labeled, and $N_{..}$ gives the total number of annotations. $\langle\rangle_F$ represents the Frobenious norm which is just the sum of the element-wise products of entries. Observed and expected coincidence matrices are each weighted by their respective discrepancies to get a scalar observed and expected discrepancy value. $\{O\}$ is computed by adding up the contributions of each item, $i$, adding up the $N_{i.}(N_{i.}-1)$ possible ordered pairs for each item and, therefore, needing to divide by $N_{i.}-1$ in order to get the item-specific total to be $N_{i.}$ (and the overall total to be $N_{..}$. Constraining the pairs to a particular ordered pair $cc'$, the first item of the pair may be one of $N_{ic}$ possible label instances, while the second one must be a *different* label instance from the set of $N_{ic'}$ labels which is why $1$ option must be removed when $c = c'$. Similarly, $E$ is computed by adding up all of the possible $N_{..}(N_{..}-1)$ pairs and, therefore, needs to be divided by $N_{..}-1$ in order to get the total to be $N_{..}$. Again, constraining the pairs to a particular ordered pair $cc'$, the first item of the pair may be one of $N_{.c}$ possible label instances, while the second one must be a *different* label instance from the set of $N_{.c'}$ labels which is why $1$ option must be removed when $c = c'$.

```
# Simplified Krippendorff's alpha via observed and expected coincidence
# Results still match https://repository.upenn.edu/asc_papers/43
```

[2] The matrix $\{N_{ic}\}$ is built in such a way that suggests that Krippendorff does not expect the same annotator to provide multiple annotations for the same item.

```
import numpy as np

def O(N):

    """Observed coincidence given item x label counts, N"""

    n, _ =N.shape

    def C(c, cP):

        return sum(

            (N[i, c] *(N[i, cP] -(1 if c ==cP else 0))) /

            (N[i, :].sum() -1)

            for i in range(n))

    return C

def E(N):

    """Expected coincidence given item x label counts, N"""

    def C(c, cP):

        return N[:, c].sum() *N[:, cP].sum() /(N[:, :].sum() -1)

    return C

def Inner(C, D, k):

    """(frobenious) Inner product of k x k matrices/func C and D"""

    return sum(C(c, cP)*D(c, cP) for c in range(k) for cP in range(k))

def alpha(N, Delta):

    """Krippendorff's alpha from item x label counts and discrepancy"""

    _, k =N.shape

    return 1 -(Inner(O(N), Delta, k) /Inner(E(N), Delta, k))

## Metrics ##

def Delta_ordinal(N):

    def Delta(c, cP):

        c, cP =sorted([c, cP])

        diff =N[:,c:(cP+1)].sum() -(N[:,c].sum() +N[:,cP].sum()) /2
        return diff **2
    return Delta
def Delta_interval(c, cP):
    return (c -cP) **2
def Delta_ratio(c, cP):
    return (((c -cP) /(c +cP)) **2) if c !=cP else 0
def Delta_nominal(c, cP):
    return 0 if c ==cP else 1
N =np.array([
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [3, 0, 0, 0, 0, 1, 0, 3, 0, 0, 2, 0],
    [0, 3, 0, 0, 4, 1, 0, 1, 4, 0, 0, 0],
    [0, 1, 4, 4, 0, 1, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 4, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0],
]).T # transpose to match paper notation and data format
assert(round(alpha(N, Delta_ordinal(N)), 3) ==0.815)
assert(round(alpha(N, Delta_interval), 3) ==0.849)
assert(round(alpha(N, Delta_ratio), 3) ==0.797)
assert(round(alpha(N, Delta_nominal), 3) ==0.743)
```

In multi-label classification, each input is labeled with a subset of possible categories as we did in Chapter 3. This can be seen as a fixed collection of binary classification tasks to be performed for each input example. Since the focus of this thesis is work that allows each dimension to be modeled in graded, non-binary ways, literature that describes how to generalize evaluation and agreement to multi-dimensional problems is relevant.

In particular, some argue that evaluating each dimension independently is insufficient for assessing the quality of annotated data or of system predictions. Bhowmick, Basu, and Mitra (2008) propose an agreement measure for the multi-label task of emotion detection. Like common chance-corrected agreement measures, the authors formulate their agreement metric, $A_m$, as a function of some observed agreement, $A_o$, and the agreement expected by chance, $A_e$:

$$A_m = \frac{A_o - A_e}{1 - A_e}$$

While we could treat the multi-label task as $C$ separate binary tasks (one for each possible label) and compute aggregate agreement by simply macro- or micro- averaging $A_m$ across those $C$ tasks, Bhowmick, Basu, and Mitra (2008) propose that averaging be over the $\binom{C}{2}$ 4-way tasks—each representing a category pair with possible labels being $[0,0],[0,1],[1,0],or[1,1]$. Their proposed measure essentially computes the aggregate Cohen's kappa but on this transformed set of tasks.[3] The idea of accounting for pairwise agreement is interesting and suggests the opportunity to evaluate higher order agreement (pair-wise or even higher) while continuing to assess more traditional unary aggregate agreement. As an example of when it may be important to still evaluate lower-order agreement as well, consider the case where two annotators annotate for exactly two properties, always agreeing on labels for first the property but never agreeing on how to label the other. Under the proposed multi-label evaluation, their observed agreement of 0 would fail to reflect the

[3] This is the basic idea of their proposal; however, they choose not to consider the $[1,0]$ case separately from the $[0,1]$ case, and the description is somewhat ambiguous about how to handle this case which would apparently lead to either over or under estimating the chance agreement, and, in some cases, leading to undefined behavior. For example, if counts from $[1,0]$ and $[0,1]$ are treated identically, then the chance agreement may be calculated as 1.0 even when actual agreement is 0. If the counts from $[1,0]$ are ignored, then the result depends on the order that a particular pair of categories is considered.

perfect agreement on the first property.[4]

One challenge of all agreement measures is determining what is an acceptable level of agreement. Many benchmarks have been proposed[5]. Beckler et al. (2018) suggest an interesting domain-adaptive approach to determining what levels of agreement are acceptable for a particular use-case. In particular, given (1) an agreement measure, (2) the final evaluation function by which computation models will be judged, and (3) some representative corpus of gold labels[6], they suggest generating several artificial annotations sets of varying qualities by adding various types of corruption [7] and plotting the corresponding agreement and objective function values. They found that an envelope on the plot emerged revealing a rationale for determining an acceptable level of agreement for the domain and objective function. It would be interesting to pursue work that considers multiple agreement measures and objective functions simultaneously. Other relevant work also exists (Beckstead, 2011; Costa-Santos et al., 2011; MORGAN et al., 2001; Hernaez, 2015; Slaug et al., 2012).

## 7.3 Aposteriori Label Aggregation with Multi-Annotator CRFs

In this section we consider the idea of *aposteriori* label aggregation—making use of annotations from multiple annotators by incorporating them within a joint model. Following

[4] It seems that this method could be combined with Krippendorff's reliability, but would require a discrepancy function $\Delta$ that can handle paired labels. A reasonable default would be to let the discrepancy of a pair be the *product* of the individual label discrepancies.

[5] The pycm package (Haghighi et al., 2018) documents many of them.

[6] They use an actual set of aggregate labels, but say this is not required.

[7] There may be productive synergy with the goals of D. Wang and Eisner (2016).

the suggestions of Passonneau and Carpenter (2014) and Hovy et al. (2013), the idea is to create a model of our various annotators so that discrepancies and correlations can be exploited by the model. Second, I consider a novel approach of identifying a particular "target annotator" and structure the model of annotation to ultimately improve our performance on that annotator while defining additional "surrogate" annotators such as the mean, max, and min annotators to provide more opportunities to learn a reliable predictor from limited data.

SPR 2.x (White et al., 2016) is an extension of the annotation effort begun by Reisinger et al. (2015)[8] that annotates data from the English Web Treebank (EWT). The EWT comes with train/dev/test splits defined that are stratified across each domain. These splits were followed by the universal dependencies project and again by the Sprl 2.x data collectors. For development purposes, we further carve off a validation set from the end of the training data. This set was formed as follows. For each genre, if there were $k$ sentences from the genre in the dev split, then the validation set will include the last $k'$ sentences from that genre in the train split where $k' >= k$ and includes full documents.

SPR 2.x allows us to consider how we might evaluate the the agreement of annotators, the difficulty of the annotation task, the reliability of the annotation protocol, a possible human bound on performance, and the mechanism by which label discrepancies should be pre-processed, ignored, or modeled.[9]

In some early work, I approached multi-annotator modeling in the context of multi-task learning, having a neural network produce a single shared sentence representation

[8] For further explanatory background, see also Chapters 1, 2, and 3 of the this thesis.

[9] White et al. (2016) include a number of interesting analyses of inter-annotator agreement on the SPR 2.x data. However, it appears that they evaluate the ability of annotators to agree on an ordering of the properties for a particular input rather than an ordering of the inputs for a particular property (which seems more germane).

which is then mapped (through a learned, differentiable function) to a lower-dimensional representation for each variable type, and then finally mapped to an annotator-specific label. I contributed to the work of Rudinger et al. (2018), but did not incorporate the idea of separate annotators as separate tasks into the final model.

In future work, I would be interested to investigate decompositional models that include both latent and observed variables about the author providing the annotations. If there is a *canonical* annotator, then I could build into the model the requirement to predict other annotations by way of exploiting correlations with a small set of labels from the canonical annotator and additional *synthetic* annotators such as the mean, max, and min labels across all annotators. Dense feature representations of knowledge about the annotator and annotation process (e.g. what time of day) could also be available at training time and jointly reasoned about at test time.

## 7.4 Apriori Label Aggregation via EASL

Work in scalar annotation, targeting ranking and scalar tasks, has led to appealing annotation protocols which I consider here for the *binary* situation frames task. In particular, we use the EASL protocol (Sakaguchi and Van Durme, 2018).

There are four ingredients in the protocol:

1. Presentation of inputs in *matches*—groups of five inputs shown together,
2. Aggregation of multiple labels for the same input (in EASL, this is formulated as the posterior mode of a Beta distribution with uniform prior, this amounts to simply taking the mean of the raw responses),

3. Incremental reassignment of groups based on updated aggregation (those with maximal posterior variance are selected and other are more likely to be selected if they have a high pairwise match score with the high variance items), and
4. The number of annotations collected for a particular input (ultimately EASL strongly favors items with scores near the middle of the range).

This section considers three aspects of annotation under the EASL protocol. The first deals with how to apply EASL when only a subset of candidate examples will receive *any* labels and when that pool of candidate examples is extremely *large* and *unbalanced* with respect to eventual scalar labels. This is likely to happen whenever the amount of data available to label greatly exceeds the resources for labeling. For example, my team confronted such a situation when seeking to label tweets from around the time and place of known disasters. The second aspect is the challenge of efficiently calibrating a scalar model by labeling a minimal amount of binary data. Finally, I consider the impact of match creation on convergence. To what degree are good matches predictable in a way that generalizes from one annotator to another? As a motivating application domain, I turn to *the situation frame identification* task.

### 7.4.1 The Situation Frames Identification Task

Despite the graded nature of semantic inference, practical applications may yet involve binary decisions. This section deals with the situation frame (SF) (Strassel, Bies, and Tracey, 2017) identification task which was created for the Darpa Lorelei program. Situation-frame detection can be approached as a multi-label binary prediction task where each document is labeled as evoking some subset of possible situation types (e.g. water shortage, terrorist incident). In many ways, this is analogous to the approach for SPR in Chapter 3. Brief descriptions of the situation types are used by annotators who manually label data that can then be used to train predictive models. The brevity of the guidelines allows annotators to quickly label many input documents without a large training investment; however, it can also negatively impact inter-annotator agreement for the tasks (and therefore, the quality and value of collected annotations).[10]

For example, the instructions for identifying water supply needs are as follows:

“You are to select message that clearly indicate a water supply need exists, where a water supply need is ‘Any situation involving water supply, including lack of water, contamination of water, etc. This type includes issues involving dietary water and agricultural water, but it does not include things like flash floods or typhoons (unless that event affects the supply of dietary or agricultural water)’.”

“Your judgment should be based on information that is asserted or strongly implied by the message, as opposed to mere speculation.”

Consider the following sentences:

1) I saw the show.

2) I’m hungry.

3) I had a nice long shower this morning.

4) I spilled the a bucket of water.

5) Water was turned off to fix plumbing.

6) We can water the lawn on Tuesdays and Saturdays.

[10] Minimal instructions can be especially important if an effort must support many languages.

7) I haven't had any water today.

8) No clean water.

(1) and (2) are clearly unrelated to water availability and (8) seems to very clearly indicate a severe water-supply need. However, annotators could easily disagree on how to appropriately label (3)-(7) in a binary fashion.

More generally, I have observed that even when it is natural to rank input sentences relative to a particular type (i.e. "rank these sentence by the degree that they indicate a water shortage"), it can still be difficult to determine the appropriate "cut-point" to distinguish positive from negative items.

To deal with such personal interpretation, it is common to have some set of inputs labeled by multiple annotators, allowing researchers to see where annotators disagree. This information is then commonly use via some combination of the following:

1. Workers are iteratively trained until they achieve sufficient agreement on held out data.

2. Annotations are only retained from those annotators achieving sufficient agreement with the consensus.

The first approaches comes at the cost of significant annotator investment by requiring additional training time and often more length annotation guidelines. The second approach also compromises efficiency of data gathering by discarding significant portions of collected data. Furthermore, both approaches result in annotations that are specific to a particular consensus or target threshold. For example, while the consensus might decide to only target extreme needs like (8) in the list above or wide-spread water needs as in (6), this would make the annotations less useful for other practical applications such as understanding the prevalence of smaller scale needs like those in (5) or (7) or possibly even (4) or (3).

In short, although data users may legitimately seek models of binary judgments, I believe that a decomposition of such a binary target task into a ranking/scalar-regression component followed by a simple binary thresholding model may accommodate lower annotator investment/overhead, more efficient use of annotation, and more reuseable annotation signal.

### 7.4.2 Ordinal Collection

In an initial annotation effort, we collected ordinal annotations for sentences provided in LoRelEI language packs. Annotators were presented with one sentence at a time and asked to determine which (if any) situation frame types (henceforth “sftypes”) were “possible, likely, or highly likely” based on the text. Sftypes were selected from a drop-down list, labeled with one of three levels via a radio button, and a ’+’ button allowed the annotator to request the opportunity to select additional types on the same sentence. The following descriptive names and examples were given as the sole explanation of the sftypes:

- Civil Unrest or Wide-spread Crime: It was also the ninth fatal attack on a teenager in the capital this week
- Elections and Politics: It marks the first time in modern French history that no major-party candidate has advanced
- Evacuation: Thousands of Americans head inland to escape Hurricane Matthew
- Food Supply: The Arctic Doomsday vault opened that stores millions of seeds from

crops around the world

- Infrastructure: The 10-year campus redevelopment project included the opening of new buildings
- Medical Assistance: A dispatcher instructed the couple to give birth on the side of the road
- Search/Rescue: We basically determined all the possible scenarios about the incident and establish what it is that that we're looking for
- Shelter: Using only his hands and materials found entirely on-site, this man built a sturdy four-walled, tile-roofed hut complete with a heated floor
- Terrorism or other Extreme Violence: Thursday's attack appeared to have been carried out by a single gunman, and the ISIS claim of responsibility was unusually swift in coming
- Utilities, Energy, or Sanitation: A power outage forced the closure of the busy station
- Water Supply: India is facing the worst drought it has seen in the last 150 years, affecting the lives of millions of people across the subcontinent

Researchers at BBN Technologies also collected labels on a subset of the same data. Their annotators similarly labeled each instance as evoking a subset of types as well as an additional list of "possible" types (i.e. the two sets gives rise to an implicit 3-way ordinal rating for each type on each input: not-possible, possible, and sure).

### 7.4.3 Scalar Annotation

The EASL annotation protocol proposed by Sakaguchi et al. allows annotators to quickly provide rich but abstract labels for input documents. Annotators are asked to directly compare a set of five examples by providing, for each, a fine-grained integer label between 0 and 100 via slider bars. These labels allow for ties but are more fine-grained than a partial ranking of the presented set of inputs since they potentially encode additional distance information in addition to ordering information. Furthermore, since the labels lie on an absolute scale, examples from different presentation sets can be roughly compared to each other.

I selected a portion of the BBN-annotated subset of LoRelEI data to further annotate with scalar labels. For each sftype, I selected those instances that received at least a "possible" label by either the BBN annotators or one of our crowd-source workers. I additionally included 100 messages from the BBN-annotated subset that received only negative labels by all annotators for all sftypes (i.e. these 100 negative examples were shared by all types whereas the positive examples may overlap across types but mostly do not). Initial EASL estimates of alpha and beta were all set to 1 which implies that that the final EASL scalars of an instance is the simple average of the scalar labels collected for that instance.[11] We required candidate mechanical turk workers to meet rigorous qualifications to be allowed to participate in the work. We annotated all eleven sftypes with 3 rounds of EASL. Subsequently, based on impressions of convergence and practical interests, we ran four additional rounds of EASL for the categories "food", "shelter", "terrorism", and "utils".

Finally, we also followed the EASL method to annotate tweets collected by Doug Jones

[11] Thanks to Chandler May for pointing this out.

et al. targeting an emergency event in Nepal. These tweets had been assigned binary labels by three different annotators leading to scores of $0, \frac{1}{3}, \frac{2}{3}$, or $1$ for the need-based sftypes: food, infra, med, search, shelter, utils, water. As with the LoRelEI language-pack data, for each sftype, I took the subset of tweets receiving at least one positive label for that type (i.e. at least a score of $\frac{1}{3}$) plus an additional 100 tweets from the collection which received scores of $0$ for all types. Treating these scores as an initial set of scalar labels, we followed the EASL protocol for three additional rounds to produce final scalar labels for the specified subset of the data.

### 7.4.4 Large-Scale Active Learning of Scalar Annotations

With the goal of obtaining a large, useful scalar-annotated corpus of tweets for each of the sftypes, we turned to large tweet collections. Specifically, we used large pools of tweets collected using keywords and time window queries to target the following eight high-profile incidents:[12]

- 2011 Volcanic explosions in Eritrea (2011-06-17T04:36:08)
- 2011 Major droughts in East Africa (2011-07-20T18:46:03)
- 2013 Major 7.7-scale earthquake in Iran near Pakistan border (2013-04-16T12:16:15)
- 2013 Overthrow of Morsi and replacement by el-Sissi in Egypt (2013-07-03T23:14:30)
- 2013 Cyclone Phallin in India (2013-10-12T17:56:24)

[12] Thanks to Ken Anderson, Mazin Hakeem, Yoshinari Fujinuma and colleagues for collecting the tweets and to Adam Poliak for computing the median time-stamps.

- 2014 Brutal crackdown on student protesters in Ethiopia (2014-05-05T03:22:20)
- 2014 Mass flooding in Turkey (2014-09-01T14:00:00)
- 2015 ISIS suicide-attack & shooting in Paris (2015-11-14T23:38:00)

For each large, incident-targeted pool of tweets, I determined the highest-traffic hour for the event (i.e. I binned tweets by hour and identified the mode bin) and selected the tweets from a 24-hour period including the mode hour, the hour previous, and the 22 hours following. To avoid a substantial amount of wasted annotation work from annotating roughly identical tweets, I chose a *canonical id* for each tweet as the earliest id of any tweet having the same text after removing trailing "t.co" urls [13] and leading retweet status indicators. [14] [15] To avoid harmful bias in trained models, the resulting dataset includes a single input per canonical id, but the particular instance is chosen uniformly-at-random from the set of items sharing a canonical id. The text seen during annotation was that of the canonical tweet.

Despite the keyword-based collection and time filtering, the remaining set of tweets was still too large for our budget [16] for annotation of 11 types, and we had early evidence of a very heavy skew toward irrelevant tweets. I suspected that the most practically useful dataset would be one that includes a roughly even balance of positive and negative examples, or even-better, a roughly even spread across human-assigned scalars.

To roughly predict human scalars, for each sftype, I used the scalar annotations described earlier to train an efficient leave-one-out cross-validated linear regression model using a pre-trained, neural sentence encoder, Infersent (Conneau et al., 2017), as features. For the

[13] `(?:\s*https?://t\.co/[a-zA-Z0-9]*\s*)*`

[14] `^(?:[rR][tT]\s+@[^: ]+:\s*)*`

[15] While this removed substantial redundancy, several very similar tweet pairs do remain in the pool.

[16] 614,797 unique tweets with 499,168 canonical tweets

given sftype and for each of the eight tweet groups (filtered to by timestamp as described above), I used the scalar predictions of the trained regression model to score all tweets as a surrogate human scalar for that sftype.

Given the surrogate scores, I turned to the challenge of selecting an evenly-spread subset of canonical tweets. First, I clipped all predicted values to fall within the $[0, 1]$ interval.[17] For each sftype and incident, I wanted to select $s = 320$ tweets that were spread as evenly as possible across the range of scores under the baseline model.

I used the following efficient heuristic strategy to accomplish this (see Figure 7.1). To select $s$ of the $n$ items from a target span of $[a, b]$, I sample a single item uniformly at random if $s = 1$ or recursively attempt to select $\frac{s}{2}$ from each half of the divided span otherwise.[18] If ever the number $n'$ of items falling into a given half interval is fewer than $\frac{s}{2}$, then all items from that half are automatically selected and $s - n'$ are selected from the other half interval. The entire procedure simply amounts to a deterministic partition of the $n$ items into $m$ subsets containing $n_i$ items for $i = 1, 2, ..., m$ and (if allocation of odd samples during recursive splitting is also considered deterministic) the allocation of the $s$ samples to those $m$ subsets. Thus each item from the $i$th subset is included with probability $\frac{s_i}{n_i}$ whereas a uniform sample of $s$ items from all $n$ would have had probability, $\frac{s}{n}$. If statistics of the original distribution are to be estimated rather than those of the biased subset, importance weights of $\frac{n_i}{s_i}$ can be associated with samples from the $i$th subset. We scored and subsampled

[17] This was necessary since the regression model erroneously predicted larger or smaller values despite never observing any at training.

[18] None of our scores were exactly on these span boundaries, and, in practice, a very small amount of random noise or arbitrary tie breaking could essentially guarantee this to always be the case. If there is an odd target number, then the odd sample is assigned uniformly at random to one of the two halves. In computing importance, weights, the analysis is greatly simplified if this tie-breaking is considered to be deterministic; indeed, one could make it clearly deterministic by following the simple heuristic that the subspan furthest from 0.5 always gets the odd sample.

```
def stat_sample(scores: list[float], s: int, a: float, b: float):
   """
   yields 's' (index, weight) corresponding to 'scores' spread between '
                                    a' and 'b'
   constraint: 0 <= a <= score[i] <= b <= 1
   """
   if s ==1 or s ==len(scores): # sample
      for index in sample(scores, size=s, replace=False):
         yield index, len(scores) /s
   else:
      mid =(a+b)/2 # partition items
      lhs, rhs =split(scored_ids, mid)

      if mid <0.5: # allocate s between two halves
         s_lhs, s_rhs =ceil(s /2), floor(s /2)
      else:
         s_lhs, s_rhs =floor(s /2), ceil(s /2)

      if s_lhs >len(lhs): # reallocate if insufficient items
         s_rhs +=s_lhs -len(lhs)
      elif s_rhs >len(rhs):
         s_lhs +=s_rhs -len(rhs)

      # recursively sample
      yield from strat_sample(lhs, s_lhs, a, mid)
      yield from strat_sample(rhs, s_rhs, mid, b)
```

**Figure 7.1.** Subsampling

separately for each of the eleven sftypes.

Given our sampled subset, we collected 2-way redundant integer labels using EASL with the simplification that both rounds included each input exactly once and groups of five were chosen uniformly at random.

### 7.4.5 Sublinear Calibration

Since the SF goal was ultimately the binary task of selecting relevant documents, a scalar predictor needs to be followed by a choice of threshold for retrieval. I will refer to this choice of threshold as “calibration” of the scalar model for the binary task. One approach to calibration is to exhaustively label a calibration set of data with binary labels; however, this seems overly expensive since there is only a single parameter to be fit. I expect a ternary-search style (like Fibonacci search) method would require significantly fewer binary labels to find the optimal calibration point.

### 7.4.6 EASL Match Optimization

The original EASL (Sakaguchi and Van Durme, 2018) was built upon previous work in *Rank Aggregation* which was rooted in efforts to determine the skill of players and to select competitive matches (Herbrich, Minka, and Graepel, 2007; Minka, Cleven, and Zaykov, 2018). It implicitly hypothesized that there was an analogue to good "matches" in the context of annotation and that *iterative refinement* of item scalars would improve such matching. The dynamic selection and grouping adds significantly to the complexity of the procedure (inhibiting research into potentially more effective rank aggregation methods for EASL) and removes the ability to annotate redundancy in parallel since it requires that the sets that are labeled in later rounds depend on the labels collected from all previous rounds. However, it is by no means clear whether we should, for example, group *similar* inputs in an effort to get more fine grained and relevant comparisons, or if we should group *dissimilar* inputs in an effort to get less noisy pairwise comparisons, if the grouping doesn't impact the results at all, or if the impact of grouping could be reduced by employing some other effort to automatically normalize annotators' use of the scale. I specifically investigated the impact of "ridit-scoring" as a way of normalizing scoring.

Ridit analysis was introduced by Bross (1958) as a simple, practical general-purpose tool. The name "ridits" was chosen with analogy to "probits" and "logits" which are both inverse cumulative density functions. The original purpose of the method of probits (Bliss, 1934) (from **prob**ability **unit**) was to offer a way to transform data that were percentages or probabilities like "the percentage of mice killed" to a more useful space where linear relationships were more easily observed and where the presence of a linear relationship could provide evidence for an underlying normal distribution. Given the standard normal density function for $P(X = x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}$ and a cumulative density $p$ with $0 \leq p \leq 1$, the probit function returns the value $x$ such that $P(X \leq x) = p$ (i.e. the point at which the area under the bell curve to the left equals the given cumulative density), also known as the z-score. The logit function similarly maps from $p$ to the point $x$ such that $P(X \leq x) = p$ under the logistic distribution $p(X = x) = \frac{1}{1+e^{-x}}$. In fact, the computation of ridit scores is actually given (without the name) in BLISS and Stevens (1937) as an intermediate step prior to the computation of probit scores when multiple observations share the same independent quantity (which is really an artifact of the imprecision of measurement). Such a step is applicable whenever the observations essentially amount to a weak ordering (allowing ties) of objects (e.g. the order in which flies die (BLISS and Stevens, 1937), or an aggregate ranking of apples with respect to flavor (Bliss, M. L. Greenwood, and E. S. White, 1956) after exposure to sprays). A weak order of objects corresponds to a discrete "empirical" cumulative distribution over ranks (or over any other monotonic property associated with the object groups) wherein the cumulative mass associated with a given position in the list includes the proportion of all objects before the given position plus half of the proportion of objects at the given position. This empirical cumulative density function, therefore, assigns a value between 0 and 1 for each object (which can then be passed through the probit function) and was later called "ridit scoring" by Bross (1958). For the case of totally ranked data, Ipsen and Jerne (1944) note that taking the probit of the (ridit) score $p = \frac{n-r+1/2}{n}$ (note the difference from the actual empirical $P(X \leq r) = \frac{n-r+1}{n}$) can be used to closely approximate what they call the "rankit" score. Rather than mapping from probabilities to z-scores, the rankit method maps from rank $r$ of $n$ to zlike-score which are the expected value of the $r$th item in a sorted list of $n$ independent draws from a standard normal distribution.

In summary, the ridit score is a surrogate cumulative density value associated with each value in a weakly ordered list (i.e. ties are allowed), so it is between 0 and 1 (in contrast to probits, logits, and rankits). The ridit score $x \in X$ is given as $r(x; X) = \frac{|x' \in X : x' < x| + \frac{1}{2}|x' \in X : x' == x|}{|X|}$. Importantly, the ridit scores are only a function of the weak ordering of the objects and do not depend on the scalars themselves, so, while the absolute magnitude of the elicited scalars allows for convenient comparison of objects across matches by a single annotator, aggregating ridits may be more appropriate that aggregating raw judgments.

Given the ability to optionally normalize scores using the ridit method, I wanted to investigate the degree to which EASL dynamic selection helps. In pilot experiments, we quickly discovered that quality of annotations varied [19] dramatically according to the quality/integrity of the annotators, which led us to collect labels from a small set of trusted annotators for the purposes of our experiments on this subject.

While much of our data-collection has been done with crowd-sourced annotators, for these experiments, I wanted to avoid confounding aspects that might arise from having to simultaneously consider the honesty of workers. For this reason, we sought a set of workers

[19] Thanks to Chandler May for many preliminary experiments on mechanical turk highlighting these issues.

that were untrained in the SF task, but trustworthy. We[20] used an available local pool of native Russian speakers to annotate a portion of the parallel Russian Lorelei data released by the LDC. I specifically focused on 920 segments that were available in Russian and with a professional English translation and which had already been annotated on a 3-way ordinal scale by annotators at BBN as described above.

There were 149 segments that had a non-zero ordinal rating for at least one of the 11 SF types. I selected all of these plus 51 additional segments selected randomly from the remaining 771 to arrive at 200 input segments. I implemented the first round of the EASL protocol by grouping these into 40 non-overlapping groups of five inputs each. For each group, I had three separate annotators provide scalar labels and I repeated this process two more times with the same workers and different (shared) random groupings.

Given this data, we can evaluate how well scores correlate (1) over all 9 annotations, (2) over all 3 shufflings for each worker, and (3) over all three workers for each shuffling. Overall Krippendorff's $\alpha$ on all raw annotations was high: $\alpha_{\text{interval}} = 0.76, \alpha_{\text{ordinal}} = 0.77, \alpha_{\text{ratio}} = 0.67$. Workers varied in their agreement across shufflings as shown in Table 7.1 (Worker 7 achieving $0.92$ intra-agreement across shuffles). Despite the scale having an absolute 0 and not allowing negative ratings, it is interesting that the interval and ordinal measurement of scale produce higher reliability numbers than ratio. The data also shows that, without any ridit normalization, intra-worker agreement is generally higher than intra-shuffle agreement and that the overall agreement roughly parallels the inter-annotator agreement. We can also see that worker-specific ridit normalization or worker+shuffle-specific normalization significantly hurt ordinal agreement but help interval and ratio agreement. Ridit normalization

[20] Thanks to Craig Harman for his "Turkle" system and for help with managing annotators.

| level | ridit | All | Shuf1 | Shuf2 | Shuf3 | Wkr11 | Wkr7 | Wkr9 |
|---|---|---|---|---|---|---|---|---|
| interval | - | 0.76 | 0.74 | 0.76 | 0.77 | 0.88 | 0.92 | 0.84 |
| | All | 0.77 | 0.74 | 0.76 | 0.79 | 0.89 | 0.89 | 0.77 |
| | HIT | 0.61 | 0.60 | 0.65 | 0.68 | 0.72 | 0.75 | 0.55 |
| | HIT+Worker | 0.67 | 0.70 | 0.73 | 0.74 | 0.69 | 0.72 | 0.62 |
| | Shuffle | 0.77 | 0.74 | 0.75 | 0.79 | 0.89 | 0.89 | 0.77 |
| | Worker | 0.79 | 0.78 | 0.78 | 0.80 | 0.89 | 0.90 | 0.77 |
| | Worker+Shuffle | 0.79 | 0.78 | 0.78 | 0.80 | 0.89 | 0.90 | 0.77 |
| ordinal | - | 0.77 | 0.74 | 0.75 | 0.79 | 0.89 | 0.90 | 0.77 |
| | All | 0.77 | 0.74 | 0.75 | 0.79 | 0.89 | 0.90 | 0.77 |
| | HIT | 0.48 | 0.56 | 0.63 | 0.62 | 0.51 | 0.58 | 0.36 |
| | HIT+Worker | 0.55 | 0.65 | 0.67 | 0.65 | 0.53 | 0.60 | 0.43 |
| | Shuffle | 0.55 | 0.74 | 0.75 | 0.79 | 0.54 | 0.65 | 0.39 |
| | Worker | 0.53 | 0.52 | 0.43 | 0.49 | 0.89 | 0.90 | 0.77 |
| | Worker+Shuffle | 0.51 | 0.52 | 0.56 | 0.49 | 0.54 | 0.66 | 0.46 |
| ratio | - | 0.67 | 0.64 | 0.66 | 0.70 | 0.86 | 0.80 | 0.68 |
| | All | 0.75 | 0.72 | 0.74 | 0.78 | 0.89 | 0.87 | 0.75 |
| | HIT | 0.53 | 0.59 | 0.64 | 0.64 | 0.60 | 0.65 | 0.43 |
| | HIT+Worker | 0.56 | 0.64 | 0.67 | 0.66 | 0.55 | 0.58 | 0.48 |
| | Shuffle | 0.75 | 0.72 | 0.74 | 0.78 | 0.89 | 0.87 | 0.75 |
| | Worker | 0.76 | 0.74 | 0.74 | 0.78 | 0.89 | 0.87 | 0.75 |
| | Worker+Shuffle | 0.76 | 0.74 | 0.75 | 0.79 | 0.88 | 0.87 | 0.75 |

**Table 7.1.** Intra-Worker Agreement in terms of Krippendorff's $\alpha$

over all of the impacts interval and ordinal agreement less but still significantly boots the ratio agreement.

EASL aggregation directs us to take the average across item labels to arrive at the consensus score. What if we randomly selected only a single label for each input (from the 9 available labels)? On average, what would the average rank correlation be? Among other things, Figure 7.2 shows the average of 100 random samples of 1, 2, ..., 9 lables (without replacement) from the 9 available labels for each item.(without replacement) of k labels for each. Figure 7.2 shows many other things as well. In addition to showing the impact of selecting $k$ random lables (without replacement) for each item, Figure 7.2 also shows the impact of other methods for selecting $k$ of the $9$ available annotations per item. For example, although it is not a viable strategy for reducing the amount of annotation needed, the line "label" shows the rank correlation to the target labels after using the $k$ *largest* labels for each item. It is interesting to note that using the highest label for each item resulted in nearly optimal correlation with the target even with only a single label per input (in otherwords, the max over labels was highly rank-correlated with the mean over labels). On the other hand, using labels in order from small to large results in artificially slow convergence to the average. The curve marked "worker mean" shows that some of the workers had better annotations than others. In particular, the worker that had the highest mean value tended to correlate best with the mean scores. Figure 7.4 shows the same results but using ridit-scores assigned separately to workers. Figure 7.3 shows the same results but using ridit-scores assigned separately to each HIT. Figure 7.5 shows the same results but using ridit-scores assigned separately to each HIT but being evaluated against a consensus formed from the average of hit-based ridit scores.

Kendall (1945) observes that there are two reasonable ways to deal with ties in rank correlation (both of which have analogs for Spearman's $\rho$ as well as the author (Kendall)'s $\tau$) and that the appropriate method depends on the situation. He credits 'Student' (i.e. William Gosset) with the representation of ties as being responsible with a reduction of variance (you get credit for ties where the other side also had ties) while he cite's Woodbury's account as giving the average over all possible consistent permutations (without changing the variance)—something that would be appropriate if there really were some total order that could/should have been hit upon but was not. We will use the 'Woodbury' formulation which simply ignores pairs in the numerator that deal with a tie either for either x or y.

As a final diagnostic of whether or matches are impactful, I considered the agreement of

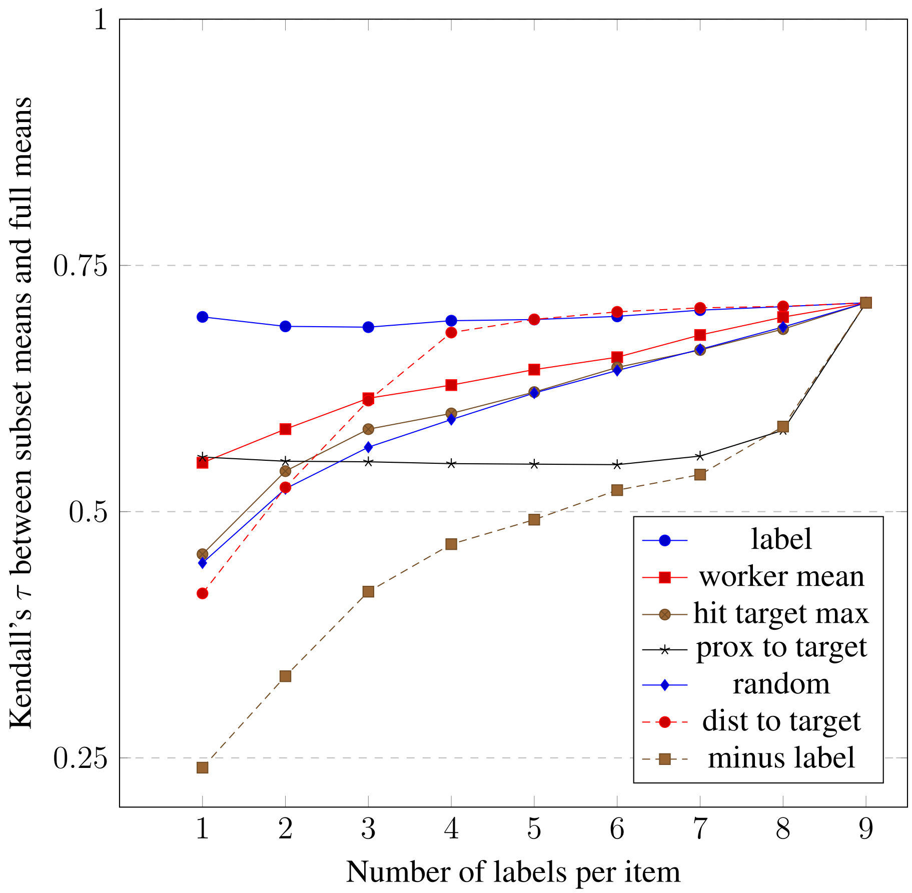


**Figure 7.2.** Convergence of Kendall's $\tau$ from Raw Scores

match rankings using only annotator-specific information to rank them. Figure 7.2 shows Krippendorff's $alpha$ coefficient for rankings of the 120 matches by each of the three workers under an ordinal measurement level. It is interesting to see that ranking matches by the minimum or median score assigned by the worker to the match does not yield high agreement of ranking (at least in part because so many matches have multiple items that score at or close to zero). However, there is high agreement on the mean score, max score, standard deviation of scores, the difference between the max and the min score, and even the total squared difference between the annotators score and the consensus mean score for each item in the match. The rows of the table correspond to various groups for ridit normalization. Normalizing by using ridit scores across each shuffle separately improves the agreement of

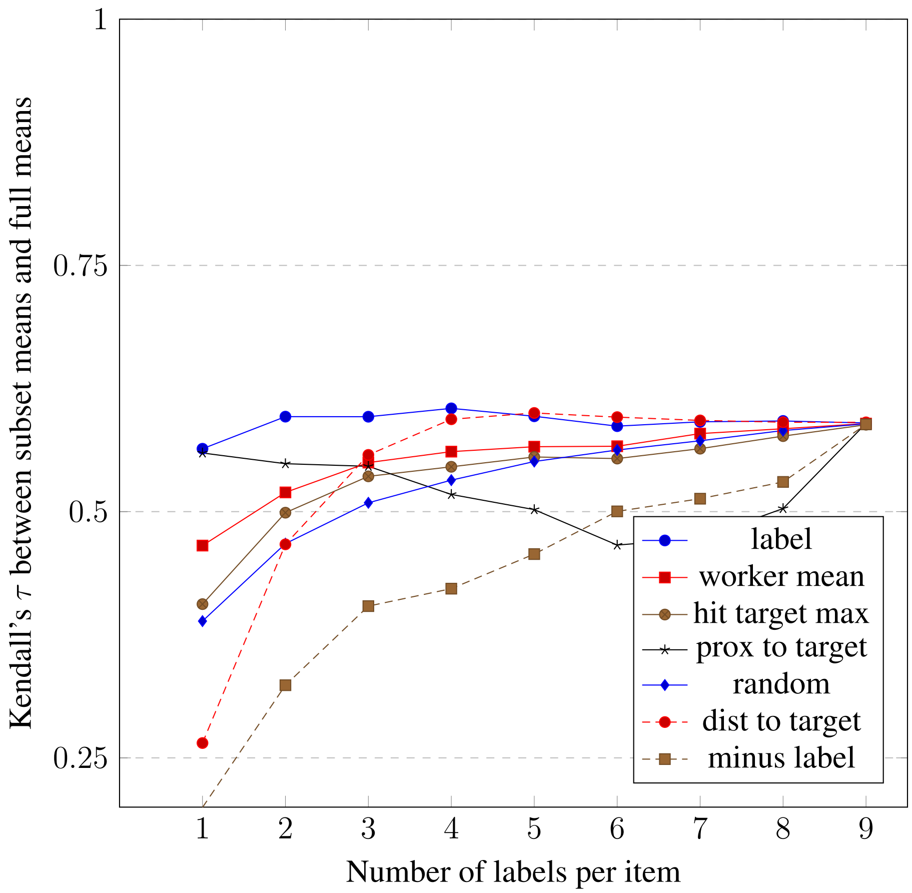


**Figure 7.3.** Convergence of Kendall's $\tau$ from Hit Ridits Against Raw Scores

worker-specific ordering of the matches.

## 7.5 Future Work

What if we didn't have an initial estimate of the EASL scores, how would we have selected instances? EASL was designed to help focus relabeling efforts on items with large label variability, but it doesn't address the case where there are many instances that will never receive any labels. If possible, we would like to maintain a current weight vector for the particular sftype based on all annotations made so far (initialized at random or by taking the difference between a fabricated clear positive example and a fabricated clear negative

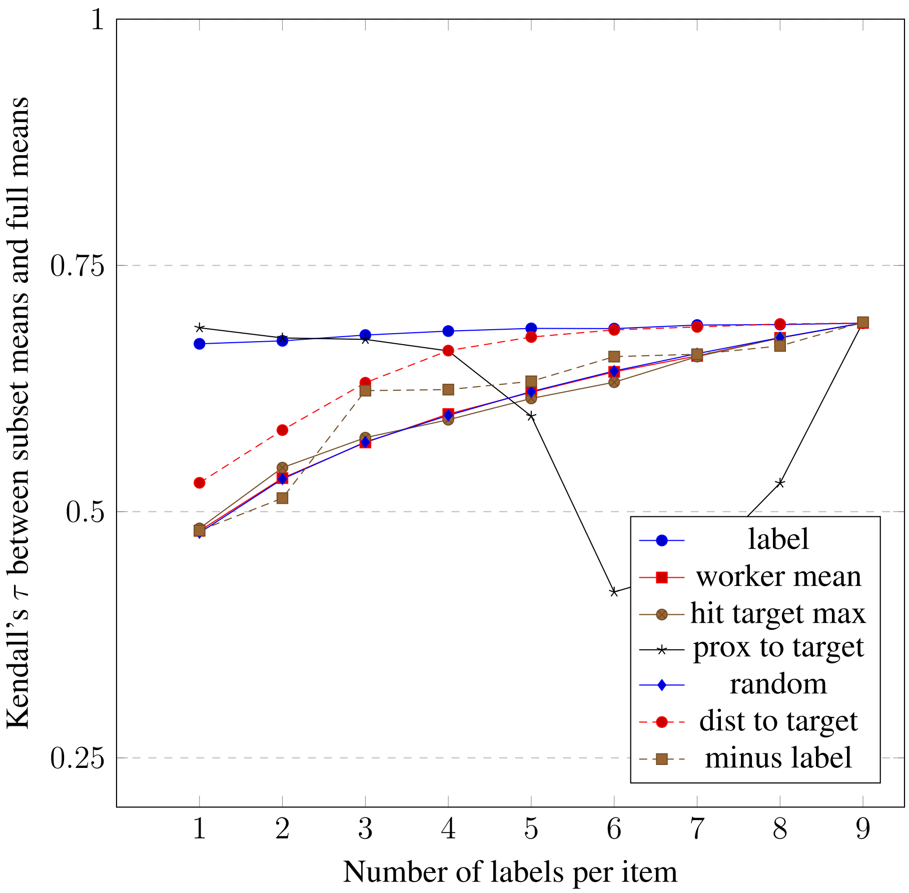


**Figure 7.4.** Convergence of Kendall's $\tau$ from Worker Ridits Against Raw Scores

example or the average of a number of contrasting pairs scaled so that the resulting score of the examples all fall roughly between 0 and 1), score all of the examples, identify the largest gaps in model score and select uniformly at random. The challenge is that it may be impractical to re-score all examples after each annotation.

One option is to do the following as an active learning strategy for this type of problem. Extract dense features for all instances once (and save them in a quickly loadable format). Each sftype will maintain a target weight vector. Throughout the process we also have a growing pool of training examples that have been labeled. We need to select five at a time to label.

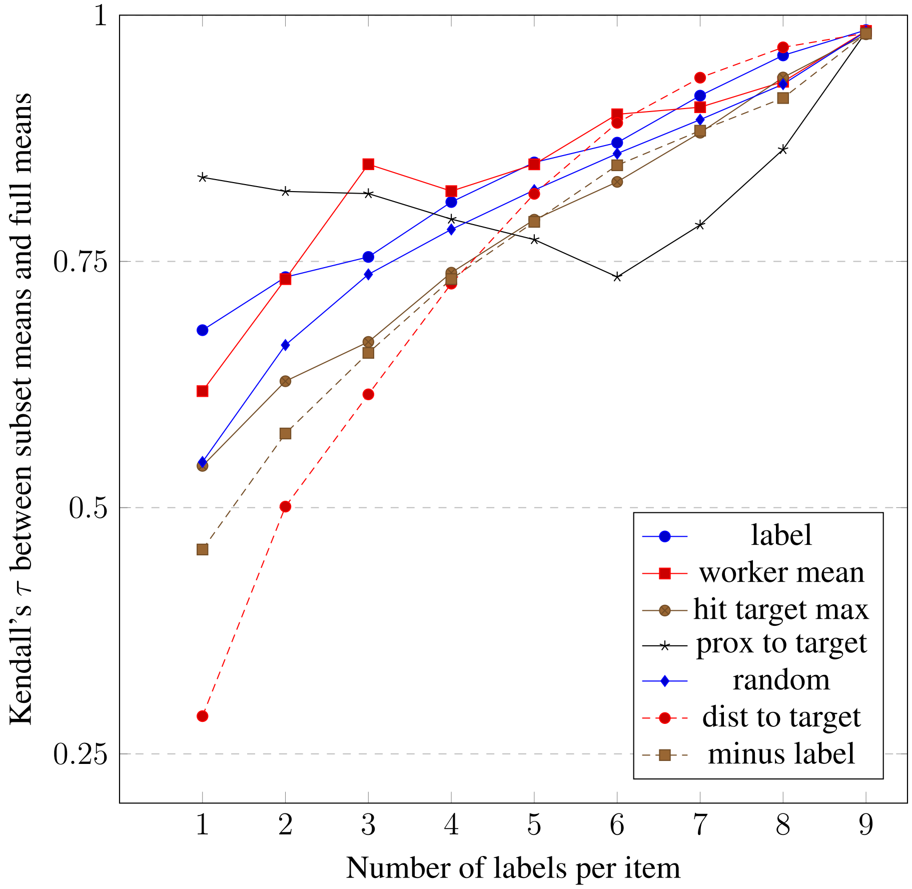


**Figure 7.5.** Convergence of Kendall's $\tau$ from Hit Ridits Against Hit Ridits

## 7.6 Conclusion

While this chapter only introduces various ideas relating to multiple annotator data and evaluation, I have presented some preliminary results that raise further questions regarding how to quick achieve meaningful agreement from annotators with minimal training.

| f | max | mean | median | min | spread | std | target |
|---|---|---|---|---|---|---|---|
| - | 0.66 | 0.75 | -0.03 | -0.48 | 0.66 | 0.70 | 0.52 |
| All | 0.66 | 0.75 | -0.03 | -0.48 | 0.66 | 0.74 | 0.47 |
| HIT | -0.02 | -0.44 | 0.24 | 0.98 | 0.55 | 0.56 | 0.43 |
| HIT+Worker | 0.24 | -0.31 | 0.25 | 0.71 | 0.56 | 0.67 | 0.56 |
| Shuffle | 0.70 | 0.75 | 0.56 | 0.05 | 0.69 | 0.74 | 0.47 |
| Worker | 0.66 | 0.74 | -0.03 | -0.48 | 0.66 | 0.75 | 0.44 |
| Worker+Shuffle | 0.67 | 0.73 | 0.30 | -0.08 | 0.66 | 0.75 | 0.43 |

**Table 7.2.** Full agreement between match rankings

# Chapter 8

# Conclusion

My overall goal in the work of this thesis has been to shed light on issues and opportunities within the decompositional approach to semantic modeling.

I began with work that related the results of a decompositional model to related non-decompositional prior work using a binary CRF for simplicity and to provide a familiar evaluation. Inference at evaluation and during training relied on loopy belief propagation as an approximation.

The ambiguous results led me to develop a python library, TorchFactors (†x), for modeling and inference which helped me investigate the joint binary task using exact inference on only a subset of the semantic properties and using a careful, two-phase hyper-parameter selection procedure. This allowed me to more clearly see the gains in held-out likelihood of the loopy model over the independent and tree-structured models. I gave a procedure for choosing a tree-structured model based on training label statistics that attained impressively close results to the loopy model. The results confirmed that clear gains in held-out likelihood did not clearly translate into clear gains in the binary F1 metrics, explaining the inconclusive results from the previous chapter. However, to clarify gains and justify the use of probabilistic joint decompositional models, I introduced the task of conditional decompositional prediction that showed that the benefits in F1 of the loopy model over the well-performing tree model grew faster as with additional available test-time observations than even the highly performant tree-based model.

Next, I investigated the finer-grained, graded aspect of the decomposition annotations, presenting a variety of standard and custom ordinal models within the CRF framework and giving higher order generalizations. Using only a fraction of the parameters, some of these models came impressively close to the nominal model with respect to held-out likelihood and other ordinal measures of evaluations, but given the many other regularizing forces available to these systems, it does not appear that there are large performance gains available from using ordinal models rather than the nominal baseline. These findings were consistent both for the single property of volition as well as a loop model involving three properties.

After clarifying the importance of joint probabilistic structure on small subsets of properties, I evaluated the impact that approximate inference would have on learning and prediction, introducing capabilities for approximate inference within my †x library. My experiments showed very little degradation from using approximate inference at test time on the six-property SPRL task, and, as expected, the tree-based models performed identically even when trained with the more efficient belief propagation inference algorithm; however, training the six-property loopy model using BP's approximation for likelihood as an objective led to significant degradation. I suggest one possible approach to alleviating the issue by encouraging convergence of messages with an explicit penalty term.

Finally, I offered thoughts, literature review, and preliminary experiments related to modeling of decompositional data from multiple annotators and using scalar annotation to potentially improve the quality of the graded signal. One interesting result from that study was that from a set of trusted workers, the max over 9 annotations for each item correlated extremely well with the mean annotation.

## 8.1 Future Work

In addition to those already highlighted throughout the thesis, there are a number of low-hanging follow-up experiments that didn't quite make it into the scope of this dissertation.

Extending the conditional evaluation of Chapter 4 to include evaluations dealing with a single reveal-order and investigating methods for choosing a single reveal-order may bear fruit for active learning of decompositional models.

In Chapter 6 it would be interesting to continue the conditional evaluation out to the full set of properties which will need to resort to sampling. Further investigation into the performance of ordinal models at very small scales of training data may show some use-cases where they would be beneficial.

There is more to be done regarding learning under approximations and improvement of approximations based on generalized BP and learned prioritization and pruning for message passing in addition to the optimal form of a non-convergence penalty during training. Training feature representations through the inference, incorporating structured, deterministic, and global factors, etc. also deserves more study.

# References

Abend, Omri and Ari Rappoport (July 2017). “The State of the Art in Semantic Representation.” In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 77–89. URL: https://www.aclweb.org/anthology/P17-1008.

Ades, Anthony E. and Mark J. Steedman (Dec. 1982). “On the order of words.” In: *Linguistics and Philosophy* 4.4, pp. 517–558. URL: https://doi.org/10.1007/BF00360804.

Akbik, Alan, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf (June 2019). “FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP.” In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 54–59. URL: https://www.aclweb.org/anthology/N19-4010.

Akbik, Alan, Duncan Blythe, and Roland Vollgraf (2018). “Contextual String Embeddings for Sequence Labeling.” In: *COLING 2018, 27th International Conference on Computational Linguistics*, pp. 1638–1649.

Anderson, J. A. (1984). “Regression and Ordered Categorical Variables.” In: *Journal of the Royal Statistical Society. Series B (Methodological)* 46.1, pp. 1–30. URL: http://www.jstor.org/stable/2345457.

Antoine, Jean-Yves, Jeanne Villaneau, and Anaïs Lefeuvre (2014). “Weighted Krippendorff’s alpha is a more reliable metrics for multi-coders ordinal annotations: experimental studies on emotion, opinion and coreference annotation.” In: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 550–559.

Aroyo, Lora (Apr. 2013). “Crowd Truth: Harnessing disagreement in crowdsourcing a relation extraction gold standard.” In: URL: https://figshare.com/articles/journal_contribution/Crowd_Truth_Harnessing_disagreement_in_crowdsourcing_a_relation_extraction_gold_standard/679997.

Aroyo, Lora and Chris Welty (2015). “Truth is a lie: Crowd truth and the seven myths of human annotation.” In: *AI Magazine* 36.1, pp. 15–24.

Artstein, Ron and Massimo Poesio (2008). “Inter-Coder Agreement for Computational Linguistics.” In: *Computational Linguistics* 34.4, pp. 555–596. eprint: https://doi.org/10.1162/coli.07-034-R2. URL: https://doi.org/10.1162/coli.07-034-R2.

Baker, Collin F., Charles J. Fillmore, and John B. Lowe (1998). “The Berkeley FrameNet Project.” In: *Proceedings of the 36th Annual Meeting of the Association for Computa-*

*tional Linguistics and 17th International Conference on Computational Linguistics - Volume 1*. ACL '98/COLING '98. Montreal, Quebec, Canada: Association for Computational Linguistics, pp. 86–90. URL: https://doi.org/10.3115/980845.980860.

Beckham, Christopher and Christopher Pal (Dec. 2016). "A simple squared-error reformulation for ordinal classification." In: *arXiv e-prints*, arXiv:1612.00775, arXiv:1612.00775. arXiv: 1612.00775 [stat.ML].

Beckham, Christopher and Christopher Pal (Aug. 2017). "Unimodal Probability Distributions for Deep Ordinal Classification." In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, pp. 411–419. URL: http://proceedings.mlr.press/v70/beckham17a.html.

Beckler, Dylan T., Zachary C. Thumser, Jonathon S. Schofield, and Paul D. Marasco (Nov. 2018). "Reliability in evaluator-based tests: using simulation-constructed models to determine contextually relevant agreement thresholds." In: *BMC Medical Research Methodology* 18.1, p. 141. URL: https://doi.org/10.1186/s12874-018-0606-7.

Beckstead, Jason W. (2011). "Agreement, reliability, and bias in measurement: Commentary on Bland and Altman (1986; 2010)." In: *International Journal of Nursing Studies* 48.1, pp. 134–135. URL: http://www.sciencedirect.com/science/article/pii/S0020748910002968.

Ben-David, Arie, Leon Sterling, and Yoh-Han Pao (1989). "Learning and classification of monotonic ordinal concepts." In: *Computational Intelligence* 5.1, pp. 45–49. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8640.1989.tb00314.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8640.1989.tb00314.x.

Ben-David, Arie, Leon Sterling, and TriDat Tran (2009). "Adding monotonicity to learning algorithms may impair their accuracy." In: *Expert Systems with Applications* 36.3, Part 2, pp. 6627–6634. URL: http://www.sciencedirect.com/science/article/pii/S0957417408005964.

Benjumeda, Marco, Concha Bielza, and Pedro Larrañaga (Sept. 2016). "Learning Tractable Multidimensional Bayesian Network Classifiers." In: *Proceedings of the Eighth International Conference on Probabilistic Graphical Models*. Ed. by Alessandro Antonucci, Giorgio Corani, and Cassio Polpo Campos. Vol. 52. Proceedings of Machine Learning Research. Lugano, Switzerland: PMLR, pp. 13–24. URL: http://proceedings.mlr.press/v52/benjumeda16.html.

Bergstra, James and Yoshua Bengio (2012). "Random search for hyper-parameter optimization." In: *Journal of Machine Learning Research* 13.1, pp. 281–305.

Bhowmick, Plaban Kumar, Anupam Basu, and Pabitra Mitra (Aug. 2008). "An Agreement Measure for Determining Inter-Annotator Reliability of Human Judgements on Affective Text." In: *Coling 2008: Proceedings of the workshop on Human Judgements in Computational Linguistics*. Manchester, UK: Coling 2008 Organizing Committee, pp. 58–65. URL: https://www.aclweb.org/anthology/W08-1209.

Björkelund, Anders, Love Hafdell, and Pierre Nugues (2009). “Multilingual semantic role labeling.” In: *Proceedings of CoNLL*, pp. 43–48.

Bliss, C. I. (1934). “The Method of Probits–A Correction.” In: *Science* 79.2053, pp. 409–410. URL: http://www.jstor.org/stable/1659363.

Bliss, C. I., Mary L. Greenwood, and Edna Sakamoto White (1956). “A Rankit Analysis of Paired Comparisons for Measuring the Effect of Sprays on Flavor.” In: *Biometrics* 12.4, pp. 381–403. URL: http://www.jstor.org/stable/3001679.

BLISS, C. I. and W. L. Stevens (1937). “The Calculation of the Time-Mortality Curve.” In: *Annals of Applied Biology* 24.4, pp. 815–852. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1744-7348.1937.tb05058.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1744-7348.1937.tb05058.x.

Bonial, Claire, Kevin Stowe, and Martha Palmer (2013). “Renewing and revising SemLink.” In: *The GenLex Workshop on Linked Data in Linguistics*.

Bottou, Léon (2012). “Stochastic gradient descent tricks.” In: *Neural Networks: Tricks of the Trade*. Springer, pp. 421–436.

Bross, Irwin D. J. (1958). “How to Use Ridit Analysis.” In: *Biometrics* 14.1, pp. 18–38. URL: http://www.jstor.org/stable/2527727.

Brown, Peter F., Peter V. Desouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai (1992). “Class-based n-gram models of natural language.” In: *Computational Linguistics* 18.4, pp. 467–479. URL: http://dl.acm.org/citation.cfm?id=176316 (visited on 08/27/2013).

Canales, Lea, Carlo Strapparava, Ester Boldrini, and Patricio Martínez-Barco (Dec. 2016). "Innovative Semi-Automatic Methodology to Annotate Emotional Corpora." In: *Proceedings of the Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media (PEOPLES)*. Osaka, Japan: The COLING 2016 Organizing Committee, pp. 91–100. URL: https://www.aclweb.org/anthology/W16-4310.

Cano, José-Ramón, Pedro Antonio Gutiérrez, Bartosz Krawczyk, Michal Woźniak, and Salvador García (2019). "Monotonic classification: An overview on algorithms, performance measures and data sets." In: *Neurocomputing* 341, pp. 168–182. URL: http://www.sciencedirect.com/science/article/pii/S0925231219302383.

Cao, Wenzhi, Vahid Mirjalili, and Sebastian Raschka (2019). "Consistent rank logits for ordinal regression with convolutional neural networks." In: *arXiv preprint arXiv:1901.07884*.

Carletta, Jean, Stephen Isard, Gwyneth Doherty-Sneddon, Amy Isard, Jacqueline C. Kowtko, and Anne H. Anderson (Mar. 1997). "The Reliability of a Dialogue Structure Coding Scheme." In: *Comput. Linguist.* 23.1, pp. 13–31. URL: http://dl.acm.org/citation.cfm?id=972684.972686.

Cheng, Weiwei, Krzysztof Dembczynski, and Eyke Hüllermeier (June 2010). "Graded Multilabel Classification: The Ordinal Case." In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. Ed. by Johannes Fürnkranz and Thorsten Joachims. Haifa, Israel: Omnipress, pp. 223–230. URL: https://icml.cc/Conferences/2010/papers/596.pdf.

Chomsky, Noam (1965). *Aspects of the Theory of Syntax*. 50th ed. Mit Press. URL: http://www.jstor.org/stable/j.ctt17kk81z.

Christensen, Rune Haubo B (2018). “Cumulative link models for ordinal regression with the R package Ordinal.” In: *Submitted in J. Stat. Software*.

Church, Alonzo (1940). “A formulation of the simple theory of types.” In: *Journal of Symbolic Logic* 5.2, pp. 56–68.

Conneau, Alexis, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes (Sept. 2017). “Supervised Learning of Universal Sentence Representations from Natural Language Inference Data.” In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 670–680. URL: https://www.aclweb.org/anthology/D17-1070.

Costa-Santos, Cristina, João Bernardes, Diogo Ayres-de-Campos, Antónia Costa, and Célia Costa (2011). “The limits of agreement and the intraclass correlation coefficient may be inconsistent in the interpretation of agreement.” In: *Journal of Clinical Epidemiology* 64.3, pp. 264–269. URL: http://www.sciencedirect.com/science/article/pii/S0895435609003643.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” In: *arXiv preprint arXiv:1810.04805*.

Dobrska, Maria, Hui Wang, and William Blackburn (Mar. 2012). “Ordinal regression with continuous pairwise preferences.” In: *International Journal of Machine Learning and Cybernetics* 3.1, pp. 59–70. URL: https://doi.org/10.1007/s13042-011-0036-x.

Domke, J. (June 2011). "Parameter learning with truncated message-passing." In: *CVPR 2011*, pp. 2937–2943.

Domke, J. (Oct. 2013). "Learning Graphical Model Parameters with Approximate Marginal Inference." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.10, pp. 2454–2467.

Dowty, David (1991). "Thematic proto-roles and argument selection." In: *Language* 67.3, pp. 547–619.

Dozat, Timothy and Christopher D. Manning (July 2018). "Simpler but More Accurate Semantic Dependency Parsing." In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 484–490. URL: https://www.aclweb.org/anthology/P18-2077.

Duchi, John, Elad Hazan, and Yoram Singer (July 2011). "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization." In: *Journal of Machine Learning Research* 12, pp. 2121–2159. URL: http://dl.acm.org/citation.cfm?id=1953048.2021068.

Dumitrache, Anca, Oana Inel, Lora Aroyo, Benjamin Timmermans, and Chris Welty (2018). "CrowdTruth 2.0: Quality metrics for crowdsourcing with disagreement." English. In: *Joint Proceedings SAD 2018 and CrowdBias 2018*. Ed. by Lora Aroyo and Anca Dumitrache. CEUR Workshop Proceedings. CEUR-WS, pp. 11–18.

Dumitrache, Anca, Oana Inel, Benjamin Timmermans, Carlos Ortiz, Robert-Jan Sips, Lora Aroyo, and Chris Welty (2018). "Empirical methodology for crowdsourcing ground truth." In: *arXiv preprint arXiv:1809.08888*.

Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin (2008). "LIBLINEAR: A Library for Large Linear Classification." In: *Journal of Machine Learning Research* 9, pp. 1871–1874.

Fathony, Rizal, Mohammad Ali Bashiri, and Brian Ziebart (2017). "Adversarial Surrogate Losses for Ordinal Regression." In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., pp. 563–573. URL: http://papers.nips.cc/paper/6659-adversarial-surrogate-losses-for-ordinal-regression.pdf.

Fernandez-Gonzalez, Pablo, Concha Bielza, and Pedro Larranaga (July 2015). "Multidimensional classifiers for neuroanatomical data." In: *ICML Workshop on Statistics, Machine Learning and Neuroscience (Stamlins 2015)*. Bertrand Thirion, Lars Kai Hansen, Sanmi Koyejo. Lille, France. URL: https://hal.inria.fr/hal-01225249.

Fernández-Navarro, F. (Aug. 2017). "A Generalized Logistic Link Function for Cumulative Link Models in Ordinal Regression." In: *Neural Processing Letters* 46.1, pp. 251–269. URL: https://doi.org/10.1007/s11063-017-9589-3.

Feyisetan, Oluwaseyi, Elena Simperl, Markus Luczak-Roesch, Ramine Tinati, and Nigel Shadbolt (2018). "An extended study of content and crowdsourcing-related performance factors in named entity annotation." In: *Semantic Web* 9.3, pp. 355–379.

Fienberg, Stephen E. (Nov. 1980). *The Analysis of Cross-Classified Categorical Data, Second Edition*. The MIT Press.

Fillmore, Charles J. (1967). "The case for case." In: URL: https://files.eric.ed.gov/fulltext/ED019631.pdf.

FitzGerald, Nicholas, Julian Michael, Luheng He, and Luke Zettlemoyer (July 2018). “Large-Scale QA-SRL Parsing.” In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 2051–2060. URL: https://www.aclweb.org/anthology/P18-1191.

Frey, Brendan J., Frank R. Kschischang, Hans-Andrea Loeliger, and Niclas Wiberg (1997). “Factor graphs and algorithms.” In: *Proceedings of the Annual Allerton Conference on Communication Control and Computing*. Vol. 35. URL: http://www.psi.toronto.edu/~psi/pubs2/1999%20and%20before/134.pdf (visited on 04/02/2015).

Fürnkranz, Johannes, Eyke Hüllermeier, and Stijn Vanderlooy (2009). “Binary Decomposition Methods for Multipartite Ranking.” In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Wray Buntine, Marko Grobelnik, Dunja Mladenić, and John Shawe-Taylor. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 359–374.

Gaag, Linda C. and Peter de Waal (Jan. 2006). “Multi-dimensional Bayesian Network Classifiers.” In: pp. 107–114.

Ganchev, Kuzman and Mark Dredze (2008). “Small statistical models by random feature mixing.” In: *Proceedings of the ACL08 HLT Workshop on Mobile Language Processing*, pp. 19–20. URL: http://www.newdesign.aclweb.org/anthology/W/W08/W08-08.pdf#page=29 (visited on 12/09/2013).

Gesmundo, Andrea, James Henderson, Paola Merlo, and Ivan Titov (2009). “A Latent Variable Model of Synchronous Syntactic-Semantic Parsing for Multiple Languages.” In:

*Proceedings of CoNLL*. Boulder, Colorado, pp. 37–42. URL: http://www.aclweb.org/anthology/W09-1205 (visited on 08/20/2015).

Gormley, Matthew R. (2015a). "Graphical Models with Structured Factors, Neural Factors, and Approximation-Aware Training." PhD thesis. Baltimore, MD: Johns Hopkins University.

Gormley, Matthew R. (2015b). "Graphical Models with Structured Factors, Neural Factors, and Approximation-Aware Training." PhD thesis. Baltimore, MD: Johns Hopkins University.

Gormley, Matthew R., Margaret Mitchell, Benjamin Van Durme, and Mark Dredze (June 2014). "Low-Resource Semantic Role Labeling." In: *Proceedings of ACL*.

Greenwood, C. and V. Farewell (1988). "A comparison of regression models for ordinal data in an analysis of transplanted-kidney function." In: *Canadian Journal of Statistics* 16.4, pp. 325–335. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.2307/3314931. URL: https://onlinelibrary.wiley.com/doi/abs/10.2307/3314931.

Gruber, Jeffrey Steven (1965). "Studies in lexical relations." PhD thesis. Massachusetts Institute of Technology.

Gutiérrez, P. A., M. Pérez-Ortiz, J. Sánchez-Monedero, F. Fernández-Navarro, and C. Hervás-Martínez (Jan. 2016). "Ordinal Regression Methods: Survey and Experimental Study." In: *IEEE Transactions on Knowledge and Data Engineering* 28.1, pp. 127–146.

Gutiérrez, Pedro Antonio and Salvador García (Aug. 2016). "Current prospects on ordinal and monotonic classification." In: *Progress in Artificial Intelligence* 5.3, pp. 171–179. URL: https://doi.org/10.1007/s13748-016-0088-y.

Gutiérrez, Pedro Antonio, M Pérez-Ortiz, Francisco Fernández-Navarro, Javier Sánchez-Monedero, and César Hervás-Martínez (2012). “An experimental study of different ordinal regression methods and measures.” In: *International Conference on Hybrid Artificial Intelligence Systems*. Springer, pp. 296–307.

Gwet, Kilem L. (2016). “Testing the Difference of Correlated Agreement Coefficients for Statistical Significance.” In: *Educational and Psychological Measurement* 76.4, pp. 609–637. eprint: https://doi.org/10.1177/0013164415596420. URL: https://doi.org/10.1177/0013164415596420.

Haghighi, Sepand, Masoomeh Jasemi, Shaahin Hessabi, and Alireza Zolanvari (May 2018). “PyCM: Multiclass confusion matrix library in Python.” In: *Journal of Open Source Software* 3.25, p. 729. URL: https://doi.org/10.21105/joss.00729.

Hajič, Jan, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang (2009). “The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages.” In: *Proceedings of CoNLL*. Association for Computational Linguistics. Boulder, Colorado, USA, pp. 1–18.

Harrell, Frank E. (2015). “Ordinal Logistic Regression.” In: *Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis*. Cham: Springer International Publishing, pp. 311–325. URL: https://doi.org/10.1007/978-3-319-19425-7_13.

He, Luheng (2018). “Annotating and Modeling Shallow Semantics Directly from Text.” PhD thesis.

He, Luheng, Mike Lewis, and Luke Zettlemoyer (Sept. 2015). “Question-Answer Driven Semantic Role Labeling: Using Natural Language to Annotate Natural Language.” In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 643–653. URL: https://www.aclweb.org/anthology/D15-1076.

Herbrich, Ralf, Tom Minka, and Thore Graepel (Jan. 2007). “TrueSkill(TM): A Bayesian Skill Rating System.” In: *Advances in Neural Information Processing Systems 20*. MIT Press, pp. 569–576. URL: https://www.microsoft.com/en-us/research/publication/trueskilltm-a-bayesian-skill-rating-system/.

Hernaez, Ruben (2015). “Reliability and agreement studies: a guide for clinical investigators.” In: *Gut* 64.7, pp. 1018–1027. eprint: https://gut.bmj.com/content/64/7/1018.full.pdf. URL: https://gut.bmj.com/content/64/7/1018.

Hoek, Jet and Merel Scholman (2017). “Evaluating discourse annotation: Some recent insights and new approaches.” In: *Proceedings of the 13th Joint ISO-ACL Workshop on Interoperable Semantic Annotation (ISA-13)*. URL: https://www.aclweb.org/anthology/W17-7401.

Hovy, Dirk, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy (June 2013). “Learning Whom to Trust with MACE.” In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, pp. 1120–1130. URL: https://www.aclweb.org/anthology/N13-1132.

Ipsen, Johs. and N. K. Jerne (1944). "Graphical Evaluation of the Distribution of Small Experimental Series." In: *Acta Pathologica Microbiologica Scandinavica* 21.2, pp. 343–361. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1699-0463.1944.tb04945.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1699-0463.1944.tb04945.x.

Jiang, Jiarong, Adam Teichert, Hal Daumé III, and Jason Eisner (June 2012). "Learned Prioritization for Trading Off Accuracy and Speed." In: *ICML Workshop on Inferning: Interactions between Inference and Learning*. 7 pages. Edinburgh. URL: http://cs.jhu.edu/~jason/papers/#jiang-et-al-2012-icmlw.

Jiang, Jiarong, Adam R Teichert, Hal Daumé III, and Jason Eisner (2012). "Learned Prioritization for Trading Off Accuracy and Speed." In: *NIPS*, pp. 1340–1348.

Jiang, Wenhan (2018). "A New Discriminative Ordinal Regression Method." In: *Procedia Computer Science* 139. 6th International Conference on Information Technology and Quantitative Management, pp. 605–612. URL: http://www.sciencedirect.com/science/article/pii/S1877050918318714.

Jianlin Cheng, Zheng Wang, and G. Pollastri (June 2008). "A neural network approach to ordinal regression." In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 1279–1284.

Johansson, Richard (2009). "Statistical Bistratal Dependency Parsing." In: *Proceedings of EMNLP*. Singapore, pp. 561–569. URL: http://www.aclweb.org/anthology/D/D09/D09-1059 (visited on 10/17/2015).

Jones, Eric, Travis Oliphant, Pearu Peterson, et al. (2001). *SciPy: Open source scientific tools for Python*. URL: http://www.scipy.org/.

Kako, Edward (2006). “Thematic role properties of subjects and objects.” In: *Cognition* 101.1, pp. 1–42.

Kendall, M. G. (1945). “The Treatment of Ties in Ranking Problems.” In: *Biometrika* 33.3, pp. 239–251. URL: http://www.jstor.org/stable/2332303.

Kim, Minyoung and Vladimir Pavlovic (2010). “Structured Output Ordinal Regression for Dynamic Facial Emotion Intensity Prediction.” In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 649–662.

Klein, Daniel (2018). “Implementing a General Framework for Assessing Interrater Agreement in Stata.” In: *The Stata Journal* 18.4, pp. 871–901. eprint: https://doi.org/10.1177/1536867X1801800408. URL: https://doi.org/10.1177/1536867X1801800408.

Kottner, Jan, Laurent Audige, Stig Brorson, Allan Donner, Byron J. Gajewski, Asbjørn Hróbjartsson, Chris Roberts, Mohamed Shoukri, and David L. Streiner (2011). “Guidelines for Reporting Reliability and Agreement Studies (GRRAS) were proposed.” In: *International Journal of Nursing Studies* 48.6, pp. 661–671. URL: http://www.sciencedirect.com/science/article/pii/S0020748911000368.

Kottner, Jan and David L. Streiner (2011). “The difference between reliability and agreement.” In: *Journal of Clinical Epidemiology* 64.6, pp. 701–702. URL: http://www.sciencedirect.com/science/article/pii/S0895435610004336.

Krippendorff, Klaus (2011). “Computing Krippendorff’s alpha-reliability.” In.

Krippendorff, Klaus (2019). *Content analysis : an introduction to its methodology*. Los Angeles: SAGE.

Kschischang, Frank R., Brendan J. Frey, and Hans-Andrea Loeliger (2001). “Factor graphs and the sum-product algorithm.” In: *IEEE Transactions on Information Theory* 47.2. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=910572 (visited on 04/02/2015).

Lafferty, John D., Andrew McCallum, and Fernando C. N. Pereira (2001). “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data.” In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML ’01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 282–289. URL: http://dl.acm.org/citation.cfm?id=645530.655813.

Laghmari, K., C. Marsala, and M. Ramdani (Oct. 2016). “Graded multi-label classification: Compromise between handling label relations and limiting error propagation.” In: *2016 11th International Conference on Intelligent Systems: Theories and Applications (SITA)*, pp. 1–6.

Li, Ling and Hsuan-tien Lin (2007). “Ordinal Regression by Extended Binary Classification.” In: *Advances in Neural Information Processing Systems 19*. Ed. by B. Schölkopf, J. C. Platt, and T. Hoffman. MIT Press, pp. 865–872. URL: http://papers.nips.cc/paper/3125-ordinal-regression-by-extended-binary-classification.pdf.

Lin, Hsuan-Tien (2008). “From ordinal ranking to binary classification.” PhD thesis. California Institute of Technology.

Liu, Yang, Yan Liu, Keith C. C. Chan, and Jing Zhang (2012). “Neighborhood Preserving Ordinal Regression.” In: *Proceedings of the 4th International Conference on Internet*

*Multimedia Computing and Service*. ICIMCS ’12. Wuhan, China: ACM, pp. 119–122. URL: http://doi.acm.org/10.1145/2382336.2382370.

Liu, Yang, Yan Liu, Shenghua Zhong, and Keith C.C. Chan (2011). “Semi-supervised Manifold Ordinal Regression for Image Ranking.” In: *Proceedings of the 19th ACM International Conference on Multimedia*. MM ’11. Scottsdale, Arizona, USA: ACM, pp. 1393–1396. URL: http://doi.acm.org/10.1145/2072298.2072023.

Lluís, Xavier, Xavier Carreras, and Lluís Màrquez (2013). “Joint arc-factored parsing of syntactic and semantic dependencies.” In: *TACL* 1, pp. 219–230.

Loper, Edward, Szu-Ting Yi, and Martha Palmer (2007). “Combining lexical resources: mapping between propbank and verbnet.” In: *Proceedings of the 7th International Workshop on Computational Linguistics*.

Lorr, Maurice and Douglas M McNair (1966). “Methods relating to evaluation of therapeutic outcome.” In: *Methods of research in psychotherapy*. Springer, pp. 573–594.

Marcus, M.P., M.A. Marcinkiewicz, and B. Santorini (1993). “Building a large annotated corpus of English: The Penn Treebank.” In: *Computational Linguistics* 19.2, p. 330.

Marneffe, Marie-Catherine de, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning (2014). “Universal Stanford Dependencies: A Cross-Linguistic Typology.” In: *Proceedings of LREC*, pp. 4585–4592. URL: http://www.lrec-conf.org/proceedings/lrec2014/pdf/1062_Paper.pdf.

McCallum, Andrew, Karl Schultz, and Sameer Singh (2009). “FACTORIE: Probabilistic Programming via Imperatively Defined Factor Graphs.” In: *Advances in Neural Information Processing Systems*. Ed. by Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A.

Culotta. Vol. 22. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2009/file/847cc55b7032108eee6dd897f3bca8a5-Paper.pdf.

McCullagh, Peter (1980). "Regression Models for Ordinal Data." In: *Journal of the Royal Statistical Society. Series B (Methodological)* 42.2, pp. 109–142. URL: http://www.jstor.org/stable/2984952.

McKinney, Wes (2010). "Data Structures for Statistical Computing in Python." In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman, pp. 51–56.

Meyers, A., R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman (2004). "The NomBank Project: An Interim Report." In: *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*. Boston, Massachusetts, USA, pp. 24–31. URL: http://www.aclweb.org/anthology/W04-2705.

Michael, Julian, Gabriel Stanovsky, Luheng He, Ido Dagan, and Luke Zettlemoyer (June 2018). "Crowdsourcing Question-Answer Meaning Representations." In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 560–568. URL: https://www.aclweb.org/anthology/N18-2089.

Minka, Tom, Ryan Cleven, and Yordan Zaykov (Mar. 2018). *TrueSkill 2: An improved Bayesian skill rating system*. Tech. rep. MSR-TR-2018-8. Microsoft. URL: https://www.microsoft.com/en-us/research/publication/trueskill-2-improved-bayesian-skill-rating-system/.

MORGAN, GEORGE A., JEFFREY A. GLINER, ROBERT J. HARMON, and Robert J. Harmon (2001). “Measurement Validity.” In: *Journal of the American Academy of Child & Adolescent Psychiatry* 40.6, pp. 729–731. URL: http://www.sciencedirect.com/science/article/pii/S0890856709604780.

Murphy, Kevin P, Yair Weiss, and Michael I Jordan (1999). “Loopy belief propagation for approximate inference: An empirical study.” In: *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., pp. 467–475.

Naradowsky, Jason, Sebastian Riedel, and David A Smith (2012). “Improving NLP through marginalization of hidden syntactic structure.” In: *Proceedings of EMNLP-CoNLL*. Association for Computational Linguistics, pp. 810–820.

Oepen, Stephan, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Zdeňka Urešová (May 2016). “Towards Comparability of Linguistic Graph Banks for Semantic Parsing.” In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Portorož, Slovenia: European Language Resources Association (ELRA), pp. 3991–3995. URL: https://www.aclweb.org/anthology/L16-1630.

Oepen, Stephan, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová (June 2015). “SemEval 2015 Task 18: Broad-Coverage Semantic Dependency Parsing.” In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Denver, Colorado: Association for Computational Linguistics, pp. 915–926. URL: https://www.aclweb.org/anthology/S15-2153.

Oepen, Stephan, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang (Aug. 2014). "SemEval 2014 Task 8: Broad-Coverage Semantic Dependency Parsing." In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland: Association for Computational Linguistics, pp. 63–72. URL: https://www.aclweb.org/anthology/S14-2008.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury (2005). "The Proposition Bank: An Annotated Corpus of Semantic Roles." In: *Computational Linguistics* 31.1, pp. 71–106. URL: http://dx.doi.org/10.1162/0891201053630264.

Passonneau, Rebecca J. and Bob Carpenter (2014). "The Benefits of a Model of Annotation." In: *Transactions of the Association for Computational Linguistics* 2, pp. 311–326. URL: https://www.aclweb.org/anthology/Q14-1025.

Paszke, Adam, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer (2017). "Automatic Differentiation in PyTorch." In: *NIPS Autodiff Workshop*. URL: https://openreview.net/forum?id=BJJsrmfCZ.

Pearl, Judea (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Petrov, Slav, Dipanjan Das, and Ryan McDonald (May 2012). "A Universal Part-of-Speech Tagset." In: *Proceedings of LREC*.

Phan, Duc-Anh, Hiroyuki Shindo, and Yuji Matsumoto (Oct. 2016). "Multiple Emotions Detection in Conversation Transcripts." In: *Proceedings of the 30th Pacific Asia Con-*

*ference on Language, Information and Computation: Oral Papers*. Seoul, South Korea, pp. 85–94. URL: https://www.aclweb.org/anthology/Y16-2006.

Plackett, R. L. (1975). “The Analysis of Permutations.” In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 24.2, pp. 193–202. URL: http://www.jstor.org/stable/2346567.

Plutchik, Robert (2001). “The Nature of Emotions: Human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice.” In: *American Scientist* 89.4, pp. 344–350. URL: http://www.jstor.org/stable/27857503.

Plutchik, Robertf (1980). “Chapter 1 - A GENERAL PSYCHOEVOLUTIONARY THEORY OF EMOTION.” In: *Theories of Emotion*. Ed. by Robert Plutchik and Henry Kellerman. Academic Press, pp. 3–33. URL: http://www.sciencedirect.com/science/article/pii/B9780125587013500077.

Raulamo-Jurvanen, Päivi, Simo Hosio, and Mika V. Mäntylä (2019). “Practitioner Evaluations on Software Testing Tools.” In: *Proceedings of the Evaluation and Assessment on Software Engineering*. EASE ’19. Copenhagen, Denmark: ACM, pp. 57–66. URL: http://doi.acm.org/10.1145/3319008.3319018.

Reisinger, Drew, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme (2015). “Semantic Proto-Roles.” In: *Transactions of the Association for Computational Linguistics* 3, pp. 475–488. URL: https://www.aclweb.org/anthology/Q15-1034.

Rennie, Jason D. M. (2005). “Loss functions for preference levels: Regression with discrete ordered labels.” In: *Proceedings of the IJCAI Multidisciplinary Workshop on Advances in Preference Handling*, pp. 180–186.

Al-Rfou, Rami, Bryan Perozzi, and Steven Skiena (Aug. 2013). “Polyglot: Distributed Word Representations for Multilingual NLP.” In: *Proceedings of CoNLL*. Sofia, Bulgaria, pp. 183–192. URL: http://www.aclweb.org/anthology/W13-3520.

Rudinger, Rachel, Adam Teichert, Ryan Culkin, Sheng Zhang, and Benjamin Van Durme (2018). “Neural Davidsonian Semantic Proto-role Labeling.” In: *Empirical Methods in Natural Language Processing (EMNLP)*. URL: http://aclweb.org/anthology/D18-1114.

Sakaguchi, Keisuke and Benjamin Van Durme (July 2018). “Efficient Online Scalar Annotation with Bounded Support.” In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 208–218. URL: https://www.aclweb.org/anthology/P18-1020.

Schubert, Lenhart (2015). “Semantic Representation.” In: URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/10051/9693.

Silva, W., J.R. Pinto, and J.S. Cardoso (2018). “A Uniform Performance Index for Ordinal Classification with Imbalanced Classes.” In: vol. 2018-July. cited By 0. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85056547543&doi=10.1109%2fIJCNN.2018.8489327&partnerID=40&md5=d87c1e3f69bdfd36d1b78b9a3bcaad35.

Slaug, Bj orn, Oliver Schilling, Tina Helle, Susanne Iwarsson, Gunilla Carlsson, and Åse Brandt (2012). “Unfolding the phenomenon of interrater agreement: a multicomponent approach for in-depth examination was proposed.” In: *Journal of Clinical Epidemiology* 65.9, pp. 1016–1025. URL: http://www.sciencedirect.com/science/article/pii/S0895435612000777.

Stanovsky, Gabriel and Ido Dagan (Oct. 2018). “Semantics as a Foreign Language.” In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 2412–2421. URL: https://www.aclweb.org/anthology/D18-1263.

Steedman, Mark (July 2019). *Combinatory Categorial Grammar*. Ed. by András Kertész, Edith Moravcsik, and Csilla Rákosi. De Gruyter Mouton. Chap. 14. URL: http://homepages.inf.ed.ac.uk/steedman/papers/ccg/moravcsik.pdf.

Stoyanov, Veselin, Alexander Ropson, and Jason Eisner (Apr. 2011). “Empirical Risk Minimization of Graphical Model Parameters Given Approximate Inference, Decoding, and Model Structure.” In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, pp. 725–733. URL: http://proceedings.mlr.press/v15/stoyanov11a.html.

Strassel, Stephanie M, Ann Bies, and Jennifer Tracey (2017). “Situational Awareness for Low Resource Languages: the LORELEI Situation Frame Annotation Task.” In: *SMERP@ ECIR*, pp. 32–41.

Sun, Haoqi, Sunil B Nagaraj, and M Brandon Westover (2018). “Predicting Ordinal Level of Sedation from the Spectrogram of Electroencephalography.” In: *2018 International Conference on Cyberworlds (CW)*. IEEE, pp. 292–295.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre (2008). “The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies.” In: *Proceedings of CoNLL*.

Sutton, Charles A. and Andrew McCallum (2012). “Improved Dynamic Schedules for Belief Propagation.” In: *CoRR* abs/1206.5291.

Teichert, Adam (2019). *What’s the difference between a Markov Random Field and a Conditional Random Field?* Cross Validated. URL:https://stats.stackexchange.com/q/421885 (version: 2019-08-12). eprint: https://stats.stackexchange.com/q/421885. URL: https://stats.stackexchange.com/q/421885.

Teichert, Adam, Adam Poliak, Benjamin Van Durme, and Matthew Gormley (2017). “Semantic Proto-Role Labeling.” In: *AAAI Conference on Artificial Intelligence (AAAI)*. URL: https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14997/14053.

Thurstone, Louis L (1927). “A law of comparative judgment.” In: *Psychological review* 34.4, p. 273.

Torra, Vicenç, Josep Domingo-Ferrer, Josep M. Mateo-Sanz, and Michael Ng (2006). “Regression for ordinal variables without underlying continuous variables.” In: *Information Sciences* 176.4. Recent advancements of fuzzy sets: theory and practice, pp. 465–474. URL: http://www.sciencedirect.com/science/article/pii/S0020025505002252.

Twomey, Niall, Rafael Poyiadzi, Callum Mann, and Raúl Santos-Rodríguez (2019). *Ordinal Regression as Structured Classification*. arXiv: 1905.13658 [cs.LG].

van der Walt, S., S. C. Colbert, and G. Varoquaux (Mar. 2011). "The NumPy Array: A Structure for Efficient Numerical Computation." In: *Computing in Science Engineering* 13.2, pp. 22–30.

Vet, H.C.W. de, C.B. Terwee, D.L. Knol, and L.M. Bouter (2006). "When to use agreement versus reliability measures." In: *Journal of Clinical Epidemiology* 59.10. cited By 786, pp. 1033–1039. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-33748604546&doi=10.1016%2fj.jclinepi.2005.10.015&partnerID=40&md5=cdf3a9abf16f716c4ed53d53bb1f4c41.

Walter, S. D., A. R. Feinstein, and C. K. Wells (Feb. 1987). "Coding Ordinal Independent Variables in Multiple Regression Analyses." In: *American Journal of Epidemiology* 125.2, pp. 319–323. eprint: http://oup.prod.sis.lan/aje/article-pdf/125/2/319/256198/125-2-319.pdf. URL: https://doi.org/10.1093/oxfordjournals.aje.a114532.

Wang, Dingquan and Jason Eisner (2016). "The Galactic Dependencies Treebanks: Getting More Data by Synthesizing New Languages." In: *Transactions of the Association for Computational Linguistics* 4, pp. 491–505. URL: https://www.aclweb.org/anthology/Q16-1035.

Weinberger, Kilian, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg (2009). "Feature hashing for large scale multitask learning." In: *Proceedings of ICML*, pp. 1113–1120. URL: http://dl.acm.org/citation.cfm?id=1553516 (visited on 01/11/2014).

Weischedel, Ralph, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston (2013). *OntoNotes Release 5.0*.

Welling, Max (2004). "On the Choice of Regions for Generalized Belief Propagation." In: *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*. UAI '04. Banff, Canada: AUAI Press, pp. 585–592.

White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme (Nov. 2016). "Universal Decompositional Semantics on Universal Dependencies." In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 1713–1723. URL: https://www.aclweb.org/anthology/D16-1177.

Whorf, Benjamin Lee (1945). "Grammatical Categories." In: *Language* 21.1, pp. 1–11. URL: http://www.jstor.org/stable/410199.

Whorf, Benjamin Lee (1965). "A Linguistic Consideration of Thinking in Primitive Communities (circa 1936)." In: *Language, Thought, and Reality: Selected Writings of Benjamin Lee Whorf*. The MIT Press, pp. 65–86.

Yedidia, J.S., W.T. Freeman, and Y. Weiss (2005). "Constructing free-energy approximations and generalized belief propagation algorithms." In: *IEEE Transactions on Information Theory* 51.7, pp. 2282–2312.

Yi, Szu-Ting, Edward Loper, and Martha Palmer (2007). "Can Semantic Roles Generalize Across Genres?" In: *Proceedings of HLT-NAACL*, pp. 548–555.

Zapirain, B., E. Agirre, and L. Màrquez (June 2008). “Robustness and Generalization of Role Sets: PropBank vs. VerbNet.” In: *Proceedings of ACL-08: HLT*. Columbus, Ohio, pp. 550–558. URL: http://www.aclweb.org/anthology/P/P08/P08-1063.

Zhao, Hai, Wenliang Chen, Chunyu Kity, and Guodong Zhou (2009). “Multilingual Dependency Learning: A Huge Feature Engineering Method to Semantic Dependency Parsing.” In: *Proceedings of CoNNL*. Boulder, Colorado, pp. 55–60. URL: http://www.aclweb.org/anthology/W09-1208.

Zillmann, Dolf (1964). “Konzept der Semantischen Aspektanalyse.” In: *Zurich: Institut fuer Kommunikationsforschung*.