Breeding Fillmore's Chickens and Hatching the Eggs: Recombining Frames and Roles in Frame-Semantic Parsing

Gosse Minnema and Malvina Nissim
Center for Language and Cognition
University of Groningen, The Netherlands
{g.f.minnema, m.nissim}@rug.nl

Abstract

Frame-semantic parsers traditionally predict predicates, frames, and semantic roles in a fixed order. This paper explores the 'chicken-or-egg' problem of interdependencies between these components theoretically and practically. We introduce a flexible BERT-based sequence labeling architecture that allows for predicting frames and roles independently from each other or combining them in several ways. Our results show that our setups can approximate more complex traditional models' performance, while allowing for a clearer view of the interdependencies between the pipeline's components, and of how frame and role prediction models make different use of BERT's layers.

1 Introduction

FrameNet (Baker et al., 2003) is a computational framework implementing the theory of frame semantics (Fillmore, 2006). At its core is the notion of *linguistic frames*, which are used both for classifying word senses and defining semantic roles. For example, in (1), "bought" is said to evoke the COMMERCE_BUY frame, and "Chuck", "some eggs", and "yesterday" instantiate its associated roles.

(1) COMMERCE_BUY
[$_{\text{Buyer}}$ Chuck] $_{\odot}$bought [$_{\text{Goods}}$ some eggs] [$_{\text{Time}}$ yesterday]

In NLP, frame-semantic parsing is the task of automatically analyzing sentences in terms of FrameNet frames and roles. It is a form of semantic role labeling (SRL) which defines semantic roles (called *frame elements*) relative to frames (Gildea and Jurafsky, 2002). Canonically (Baker et al., 2007), frame-semantic parsing has been split up into a three-component pipeline: `targetID` (find frame-evoking predicates), then `frameID` (map each predicate to a frame), and lastly `argID` (given a predicate-frame pair, find and label its arguments). Some recent systems, such as the LSTM-based Open-SESAME and (Swayamdipta et al., 2017) or the classical-statistical SEMAFOR (Das et al., 2014), implement the full pipeline, but with a strong focus specifically on `argID`. Other models implement some subset of the components (Tan, 2007; Hartmann et al., 2017; Yang and Mitchell, 2017; Peng et al., 2018), while still implicitly adopting the pipeline's philosophy.[1] However, little focus has been given to frame-semantic parsing as an *end-to-end* task, which entails not only implementing the separate components of the pipeline, but also looking at their interdependencies.

We highlight such interdependencies from a theoretical perspective, and investigate them empirically. Specifically, we propose a BERT-based (Devlin et al., 2019) sequence labeling system that allows for exploring frame and role prediction independently, sequentially, or jointly. Our results (i) suggest that the traditional pipeline is meaningful but only one of several viable approaches to end-to-end SRL, (ii) highlight the importance of the `frameID` component, and (iii) show that, despite their interdependence, frame and role prediction need different kinds of linguistic information.

Contributions The main contributions of this paper are the following:

- We identify theoretical and practical challenges in the traditional FrameNet SRL pipeline (§2);
- We introduce a flexible, BERT-based sequence-labeling architecture, and experiment with predicting parts of the pipeline separately (§3);
- We explore four methods for re-composing an end-to-end system (§4);

[1] Yang and Mitchell (2017) and Peng et al. (2018) learn frames and arguments jointly, but still need `targetID` as a separate step.

- Through two evaluation metrics, we empirically show the relative contribution of the single components and their reciprocal impact (§5-6).

All of our source code and instructions for how to reproduce the experiments is publicly available at `https://gitlab.com/gosseminnema/bert-for-framenet`.

2 On pipelines, chickens, and eggs

According to Fillmore (2006), an essential feature of a frame is that it is "any system of concepts related in such a way that to understand any one of them you have to understand the whole structure in which it fits." In particular, linguistic frames are systems of semantic roles, possible predicates, and other semantic information. In this section, we discuss the relationship between these concepts in the context of frame-semantic parsing and highlight interdependencies between the various components.

2.1 Challenges for parsers

The following artificial examples display some of the challenges that frame-semantic parsers face:

(2) SELF_MOTION
[$_{\text{Self\_mover}}$ Angela] $\odot$ran [$_{\text{Goal}}$ to school]

(3) FLUIDIC_MOTION
[$_{\text{Fluid}}$ A tear] $\odot$ran [$_{\text{Path}}$ down my cheek]

(4) EXPEND_RESOURCE
[$_{\text{Agent}}$ We] $\odot$ran $\odot$out [$_{\text{Resource}}$ of cookies]

(5) $\emptyset$
His girlfriend ran him home.[2]

In each example, the predicate contains "ran", but used in different frames. In (2) and (3), the predicate is the verb "run", but used in two different senses (running of a person vs. running of a liquid), corresponding to two different frames. Here, the main parsing challenge is resolving this ambiguity and choosing the correct frame (`frameID`). By contrast, in (4), the predicate is "run out". This complex verb is not ambiguous, so the main challenge in this sentence would be `targetID` (i.e. identifying that the target consists of the two tokens "ran" and "out"). Similarly, in (5), "run" is used in a sense not listed in FrameNet, so the challenge here is to make sure nothing is tagged at all.

[2]See sense #14 of "run" in `https://www.oxfordlearnersdictionaries.com/definition/english/run_1?q=run`

The *roles-make-the-frame* problem In (2-3), given the target ("ran"), the task is to find the correct frame and its corresponding roles. In the traditional pipeline, we would do this by first predicting a frame, and then labeling the dependents of "ran" with roles from this frame. However, the question is what kinds of patterns a frame-finding model needs to learn in order to be successful. It is clearly not sufficient to learn a one-to-one mapping between word forms and frames, not just because of known ambiguous cases ("Angela runs" vs. "a tear runs"), but also because of gaps in FrameNet that conceal unknown ambiguities, such as in (5).

To distinguish between "ran" in (2) and (3), a model has to to take into account the sentential context in some way, which is exactly what LSTM-based models or BERT-based models can do. But what kind of contextual information exactly do we need? SELF_MOTION and FLUIDIC_MOTION have a very similar syntax and semantics, the crucial difference being the semantic category of the "mover". Concretely, this means that in (2-3), we would benefit from recognizing that "Angela" denotes an animate entity while "a tear" denotes a fluid. Doing so would amount to doing partial semantic role labeling, since we are looking at the predicate's syntactic arguments and their semantic properties, which is exactly the information an `argID` model needs to tag "Angela" with "Self_mover" and "a tear" with "Fluid". While it is possible to use contextual information without knowledge of dependency structure (perhaps simple co-occurrence is enough), we hypothesize that such knowledge would be helpful, and thus, that doing `frameID` and `argID` simultaneously, or even predicting `frameID` after `argID`.

The *frames-make-the-targets* problem In the literature, `targetID` has received even less attention than `frameID` — all models we are aware of use gold `targetID` inputs — but is crucial to the success of any end-to-end model. Theoretically speaking, the `targetID` problem is less interesting than `frameID`: since as almost any content word can evoke a frame, assuming a fully complete FrameNet (containing all possible predicates), doing `targetID` would amount to a (simplified) POS-tagging task where content words are labeled as "yes", and (most) function words as "no".

However, in practice, FrameNet is far from complete, so that doing `targetID` means identifying all wordforms that correspond to some predicate evoking a frame present in FrameNet, making `targetID` dependent on `frameID`.[3] For example, to find the target in (2-3), it would suffice to lemmatize "ran" to "run", and check if "run" is listed under any FrameNet frames. But this strategy would fail in (4-5): in those cases, 'ran' is not the full target, but either only a part of it (4), or not at all (5). In order to predict this, we would need to recognize that "run out" is part of the EXPEND_RESOURCE frame, and that "run someone somewhere" is a different sense of "run" that does not match either FLUIDIC_MOTION or SELF_MOTION. Hence, `targetID` seems to presuppose (partial) `frameID` in some cases.

2.2 Pipelines: NLP vs. SRL

The type of problem that we identified in this section is not unique to frame-semantic parsing but also occurs in the standard NLP pipeline of tokenization, POS-tagging, lemmatization, etc. For example, for POS-tagging "run" as either a verb or a noun (as in "we run" vs. "a long run"), one (theoretically speaking) needs access to dependency information (i.e. is there a subject, adjectival modification, etc.). Conversely, dependency parsing benefits from access to POS tags. This would imply that a traditional pipeline might need a lot of redudancy; e.g., a perfect POS-tagging model would also learn some dependency parsing. For (amongst others) this reason, the problem of pipelines versus joint prediction has been extensively studied in NLP in general and SRL in particular. For example, Toutanova et al. (2005) found that predicting all PropBank semantic roles together produced better results than predicting each role separately, Finkel and Manning (2009) proposed a joint model for syntactic parsing and named entity recognition as an alternative to separate prediction or a pipeline-based approach, and He et al. (2018) propose predicting PropBank predicates and semantic roles together instead of sequentially. However, as far as we are aware, no work so far has systematically addressed the frame semantic parsing pipeline and the possible ways for arranging its different components.

In modern NLP, traditional pipelines have largely been replaced by neural models performing several tasks at once. However, a line of work initiated by Tenney et al. (2019); Jawahar et al. (2019) shows that neural models like BERT implicitly learn to reproduce the classical NLP pipeline, with different layers specializing in specific components of the pipeline, and the possibility for later layers to dynamically resolve ambiguities found in earlier layers. For the BERT-based models we propose, we study the relationship between different layers and the traditional FrameNet pipeline (cf. §6.2).

Input	Sequence labels	Frame structures
Boris	R:Self_mover|R:Sound_source	⟨⟨MAKE_NOISE, ⟨screamed⟩⟩,
screamed	F:Make_noise	{⟨Sound_source, ⟨Boris⟩⟩}⟩
and		⟨⟨SELF_MOTION, ⟨ran⟩⟩,
ran	F:Self_motion	{⟨Self_mover,⟨Boris⟩⟩,
home	R:Direction	⟨Direction, ⟨home⟩⟩}⟩

Figure 1: Frame structures and sequence labels (N.B.: color added for illustrative purposes only)

3 Dissecting the pipeline

We argued that the different components of the frame-semantic parsing task are mutually dependent on each other. Here, we take a more practical view and re-define the parsing problem in a way that allows for experimenting with individual parts of the pipeline and different combinations of them.

3.1 Strip the parser: just sequence labels

For our purposes, a crucial limitation of existing frame-semantic parsing models is that they are relatively complex and not very flexible: the different components have to be executed in a fixed order and depend on each other in a fixed way, leaving no room for experimenting with different orders or alternative ways to combine the components.

By contrast, we propose a maximally flexible architecture by redefining frame-semantic parsing as a sequence labeling task: given a tokenized sentence $S = \langle t_1, \ldots, t_n \rangle$, we predict a frame label sequence $FL = \langle l_1, \ldots, l_n \rangle$, where every $l_i \in (FID \cup \{\emptyset\}) \times 2^{AID}$ is a pair of zero or one frame labels in $FID = \{F_{\text{Abandonment}}, \ldots, F_{\text{Worry}}\}$ and zero or more role labels in $AID = \{A_{\text{Abandonment@Agent}}, \ldots, A_{\text{Worry@Result}}\}$. Note that there can be more than one frame in every sentence, and the spans of different roles can overlap. This is illustrated in Figure 1: *Boris* has two RID labels, each of which is associated to a different frame (Self_mover belongs with SELF_MOTION, while Sound_source belongs to MAKE_NOISE.

This problem definition comprises several simplifications. First of all, we integrate `targetID` and `frameID` into a single component. Moreover,

[3]It also makes the task somewhat arbitrary (since it depends on what happens to be annotated in FrameNet), leading some researchers to ignore the problem altogether (Das, 2014).

we 'flatten' the role labels, discarding predicate-role dependency information, and assume that most of this information can be recovered during post-processing (see §5.2). We further simplify the role labels by removing frame names from argument labels, as in $AID' = \{A_{\text{Agent}}, \dots, A_{\text{Result}}\}$. While this complicates recovering structural information, it also greatly condenses the label space and might improve generalization across frames: many frames share roles with identical names (e.g., Time, Location, or Agent), which we assume are likely to share at least some semantic properties. It should be noted that this assumption is not trivial, given that there is a long and controversial literature on the generalizability of semantic (proto-)roles (Reisinger et al., 2015); we will make it here nonetheless, especially since initial experiments on the development set showed a clear advantage of removing frame names from argument labels.

We implement our architecture using a BERT-based sequence labeler: given a sentence, we tokenize it into byte-pairs, compute BERT embeddings for every token, feed these (one-by-one) to a simple feed-forward neural network, and predict a label representation. By having BERT handle all preprocessing, we avoid making design choices (e.g. POS-tagging, dependency parsing) that can have a large impact on performance (cf. Kabbach et al., 2018), and make our approach easier to adapt to other languages and datasets.

3.2 Strip the tasks: just frames, just roles

Having maximally 'stripped down' the architecture of our parsing model, we can now define the two most basic tasks: frame prediction (equivalent to `targetID` plus `frameID` in the traditional pipeline), or role prediction (equivalent to `argID`, but without needing frames as input). We can then perform the tasks separately, but also jointly, or combine them in any desired way.

FRAMESONLY The first basic task is predicting, given a token, whether this token 'evokes' a FrameNet frame, and if so, which one. We experiment with two types of label representation settings: **Sparse**, which represents each frame (and the empty symbol) as a one-hot vector, while **Embedding** defines dense embeddings for frames. The embedding of a frame F is defined as the centroid of the embeddings of all predicates in F, which in turn are taken from a pre-trained GloVe

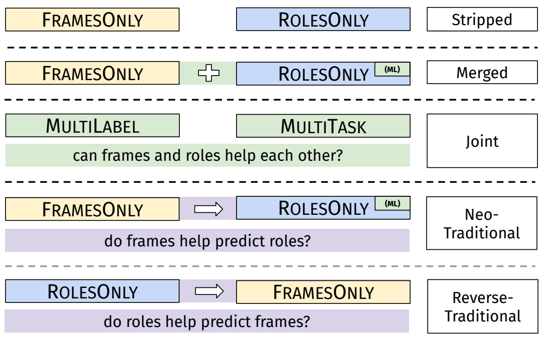

Figure 2: Overview of possible end-to-end systems *(N.B.: boxes marked with (ML) use role predictions from* MULTILABEL*)*

model (Pennington et al., 2014).[4] This is very similar to the approach taken by Alhoshan et al. (2019). Prediction is done by regressing to the frame embedding space, and selecting the frame with the smallest cosine distance to the predicted embedding. The empty symbol is predicted if the cosine similarity to the best frame is below a threshold t_f.

ROLESONLY The other basic task predicts zero or more bare role labels for every token in the input. These labels are encoded in a binary vector that represents which roles are active for a given token. During decoding, tokens with an activation value exceeding a threshold t_r are kept as the final output.

4 Re-composing an end-to-end system

Having defined a basic setup for experimenting with predicting frames and roles alone, we can now design experiments for investigating any interactions between frames and roles. Figure 2 provides an overview of the possible ways for combining the FRAMESONLY and ROLESONLY models: simply merging the outputs, predicting the two tasks jointly, or using a sequential pipeline.

4.1 Do-it-together: multilabel or multitask

Given the overlap between the frame and role prediction tasks, we test whether predicting frames and roles jointly might help the two models mutually inform each other and learn more efficiently.

Joint(MULTILABEL) The first 'joint' approach is to predict, for every token in the input, a binary vector representing any frame target, as well as any role labels carried by the token. Hence, there is only one decoder and all parameters are shared.

[4]We use the model `glove.42B.300d` from `https://nlp.stanford.edu/projects/glove/`.

Joint(MultiTask) As an alternative, we also try a setup with separate decoders for roles and frames, without any shared parameters (except for the BERT encoder). Backpropagation is done based on a weighted sum of the losses of the two decoders, where the 'loss weights' are learned.

4.2 Do-it-sequentially: what comes first?

Neo-traditional In the *Neo-traditional* experiment, we test the traditional pipeline structure: i.e., learning frames first, and using these to (explicitly) inform role learning. In order to do this, we make two modifications to RolesOnly: 1) we split the target role labels by frame, i.e. we ask the model to predict only one frame's roles at any given time, and 2) a representation of the 'active' frame is concatenated to the BERT embeddings as input to the model. This representation could be either **Sparse** or **Embedding** (see above). After role prediction, any roles that did not match the frame inputs are filtered out, and the predictions are merged with the frame model's output.

Neo-traditional+MultiLabel Following preliminary results, we repeat the experiment using MultiLabel instead of RolesOnly. In a final merging step, we keep all role predictions from MultiLabel and any frame predictions that do not clash with the outputs of FramesOnly.

Reverse-traditional In this setup, we invert the traditional pipeline: given a sentence, we first predict role labels (using RolesOnly), which are then used as input for the frame prediction model.[5]

4.3 Do-it-separately: copy-and-paste

Finally, we tried an approach assuming no interaction between frame and role prediction at all.

Merged In the *Merged* experiment, we simply merge the outputs of FramesOnly and RolesOnly. In this scenario, both models are completely independent, without any possibility for frames and roles to inform each other.

Merged+MultiLabel Based on initial results showing that MultiLabel beats RolesOnly on roles while FramesOnly wins on frames, we also experiment with simply merging the output of these two 'winning' models.

[5]Other setups, e.g. using MultiLabel for role predictions, might give better performance, but would obfuscate the effect of predicting roles before frames.

5 Evaluation: tokens vs. structures

Since our setup diverges significantly from previous systems, testing our models is not trivial. Here, we propose two evaluation methods: a token-based metric that can directly score our models' output (§5.1), and an algorithm for 'recovering' full frame structures that can be checked using the standard SemEval 2007 method (Baker et al., 2007) (§5.2).

5.1 Sequence-label evaluation

The simplest way of evaluating our models' performance is to simply count the number of correct frame and role labels per token. We compute this given a token sequence $\langle t_1, \ldots, t_n \rangle$, a sequence of gold labels $\langle G_1, \ldots, G_n \rangle$ and a sequence of predicted labels $\langle P_1, \ldots, P_n \rangle$, where every G_i and P_i is either a set of frame or role labels, or the empty label $\{\emptyset\}$. We can now define: $\textit{true\_positive} = \sum_{i=1}^{n} |P_i \cap G_i|$, $\textit{false\_positive} = \sum_{i=1}^{n} |P_i \setminus G_i|$, and $\textit{false\_negative} = \sum_{i=1}^{n} |G_i \setminus P_i|$. Finally, we calculate micro-averaged precision, recall, and F-scores in the usual way.

Consistency scoring A limitation of our sequence-labeling approach is that there are no explicit constraints on the predicted role labels and it is not guaranteed that the set of role predictions for a given sentence will be compatible with the set of frame predictions. Hence, we need to evaluate not just the respective accuracy, but also the mutual consistency, of predicted roles and frames. We define this as $\sum_{s \in S} \sum_{t \in \textit{tok}(s)} |\{r \in R_{s,t} | r \in \textit{allowed}(F_s)\}|$, where S is the set of sentences in the evaluation set, $\textit{tok}(s)$ returns the sequence of tokens in a sentence, $R_{s,t}$ is the set of predicted role labels for a particular token, F_s is the set of all predicted frame labels in the sentence, and $\textit{allowed}(F)$ returns the set of role labels that are consistent with a particular set of frame labels. For example, $\textit{allowed}(\{\text{Killing}, \text{Using}\})$ gives $\{\text{Killer}, \text{Victim}, \ldots, \text{Agent}, \ldots\}$. The number of consistent roles is then divided by the total number of predicted roles $\sum_s \sum_t |r \in R_{s,t}|$ to yield a global consistency score.

5.2 Recovering frame structures

For comparing our models to existing work in frame-semantic parsing, and validating the assumptions underlying our sequence-labeling setup, we need to recover full frame structures from the output of our models. Formally,

	DEV						TEST					
	frames			roles			frames			roles		
Experiment	**R**	**P**	**F**	**R**	**P**	**F**	**R**	**P**	**F**	**R**	**P**	**F**
Open-SESAME	*0.66*	*0.68*	*0.67*	*0.41*	*0.54*	*0.47*	*0.58*	*0.62*	*0.60*	*0.38*	*0.41*	*0.39*
Joint(MULTILABEL)	0.58	0.65	0.61	0.39	0.48	0.43	0.55	0.44	0.49	0.36	0.27	0.31
Joint(MULTITASK)	0.69	0.73	0.71	0.24	0.35	0.28	0.65	0.49	0.56	0.24	0.21	0.21
Neo-traditional(MULTILABEL)*†	0.66	0.73	0.69	0.41	0.54	0.47	0.64	0.51	0.57	0.40	0.30	0.34
Reverse-traditional(ROLESONLY)‡	0.68	0.72	0.70	0.32	0.46	0.38	0.65	0.49	0.56	0.31	0.27	0.28
Merged(MULTILABEL)*	0.72	0.68	0.70	0.39	0.49	0.43	0.69	0.48	0.57	0.36	0.28	0.31
Stripped(FRAMESONLY, Embedding)	0.68	0.69	0.69	-	-	-	0.65	0.46	0.54	-	-	-
Stripped(FRAMESONLY, Sparse)	0.65	0.75	0.70	-	-	-	0.63	0.52	0.57	-	-	-
Stripped(ROLESONLY)	-	-	-	0.32	0.46	0.38	-	-	-	0.31	0.27	0.28

Table 1: Sequence labeling scores (avg. over three runs). *N.B: *For brevity reasons, for Merged and Neo-traditional, we only give results for the* MULTILABEL *setting, which performs better on role prediction than* ROLESONLY. † *Results on Neo-traditional are using Sparse frame inputs.* ‡ *Results on Reverse-traditional are using Sparse frame outputs.*

given a tokenized sentence $\langle t_1, \ldots, t_n \rangle$, and a sequence $\langle l_1, \ldots l_n \rangle$ of frame and role labels, we want to find the set of frame structures $\{\langle TI_1, RI_1 \rangle, \ldots, \langle TI_n, RI_n \rangle\}$. Here, every target instance $TI_i = \langle FT_i, \langle t_j, \ldots, t_k \rangle \rangle$ is a pairing of a frame type $FT \in \{F_{\text{Abandonment}}, \ldots F_{\text{Worry}}\}$ and a sequence of tokens containing the lexical material that evokes the frame. Similarly, we define every role instance $RI_i = \{\langle RT_{i_1}, \langle t_{j_1}, \ldots t_{k_1} \rangle \rangle, \ldots, \langle RT_{i_n}, \langle t_{j_n}, \ldots, t_{k_n} \rangle \rangle\}$ as a set of pairs of role types $RT \in \{A_{\text{Abandonment@Agent}}, \ldots, A_{\text{Worry@Result}}\}$ and token spans instantiating these role types. See Figure 1 (§3) for an example sentence with sequence labels and corresponding frame structures.

Recovery algorithm We propose a simple rule-based approach. First, we find the set of target instances in the sentence, and the corresponding set of frame types.[6] Next, we find the set of (bare) role labels that can be associated to each of the predicted frame types, e.g. WORRY $\mapsto$ {Experiencer, . . . , Result}. Next, for each of the the predicted role spans $\langle t_i, \ldots, t_j \rangle$ in the sentence, we find all of the compatible frame target instances. If there is more than one compatible target, we select the target that is closest in the sentence to the role span. Note that our algorithm would miss cases of more than one frame instance 'sharing' a role (i.e. all having a role with the same label and span), but we assume that such cases are rare. In cases where it is already known which role labels are associated to which frame types (i.e., in the *Neo-traditional* setup), we allow the algorithm to take this information into account, but we found that this has little impact on performance.

[6]If more than one frame target label is predicted for a given token, we only keep the label with the highest probability.

SemEval'07 scoring Having recovered the set of predicted frame structures, we can evaluate our models using the standard SemEval 2007 scoring method (Baker et al., 2007). During evaluation on the development set, we noticed that our models frequently seem to make minor mistakes on role spans (i.e. erroneously missing or including an extra token). Since the SemEval script does not take into account partially matching role spans, we propose a modification to the script that gives partial credit for these role spans, and report this in addition to the scores from the original script.

6 Experiments

6.1 Setups

All experiments were run using a pre-trained `bert-base-cased` model, fine-tuned with a simple feedforward network decoder. Loss functions depend on the setup: we optimize Mean Squared Error Loss for ROLESONLY and MULTILABEL, Sequence Cross-Entropy Loss for FRAMESONLY/Sparse, and Cosine Embedding Loss for FRAMESONLY/Embedding. We found best performance using Adam optimization (Kingma and Ba, 2014) with $lr = 5e^{-5}$, training for 12 epochs, with a single hidden layer of size 1000 in the decoder. Unless specified otherwise, the BERT embeddings are an automatically weighted sum ("Scalar Mix") of BERT's hidden layers. For implementation, we used AllenNLP (Gardner et al., 2017) and PyTorch (Paszke et al.,

	DEV						TEST					
	strict			modified			strict			modified		
Experiment	**R**	**P**	**F**	**R**	**P**	**F**	**R**	**P**	**F**	**R**	**P**	**F**
Open-SESAME (true)	*0.47*	*0.52*	*0.49*	*0.51*	*0.57*	*0.54*	*0.44*	*0.50*	*0.47*	*0.47*	*0.53*	*0.50*
Open-SESAME (recovered)	*0.46*	*0.51*	*0.48*	*0.50*	*0.56*	*0.53*	*0.43*	*0.49*	*0.46*	*0.46*	*0.53*	*0.49*
Joint(MULTILABEL)	0.31	0.40	0.35	0.37	0.48	0.42	0.32	0.42	0.36	0.36	0.48	0.41
Joint(MULTITASK)	0.34	0.44	0.38	0.38	0.50	0.43	0.36	0.43	0.39	0.40	0.48	0.43
Neo-traditional(MULTILABEL)*†	0.33	0.38	0.35	0.42	0.49	0.45	0.34	0.37	0.35	0.42	0.46	0.44
Reverse-traditional(ROLESONLY)‡	0.33	0.48	0.39	0.38	0.55	0.45	0.34	0.47	0.40	0.38	0.53	0.45
Merged(MULTILABEL)*	0.37	0.44	0.40	0.44	0.52	0.47	0.40	0.44	0.42	0.45	0.50	0.47

Table 2: SemEval'07 scores (avg. over three runs)

2019). All models were trained and tested on the standard FrameNet 1.7 fulltext corpus (see Appendix B for more details on the data).

While our main aim remains a deeper understanding of the components of frame-semantic parsing and their interdependencies, we still need to put our scores into perspective and legitimize our sequence labeling approach. Thus, we took Open-SESAME, the only existing, open-source model that we are aware of that is capable of producing end-to-end predictions, as our baseline.[7] We used default settings (i.e., without scaffolding and ensembling) for better comparability with our own models. Hence, note that the results reported here for Open-SESAME are *not* the state-of-the-art results reported by Swayamdipta et al. (2017).

6.2 Results

Sequence labeling Table 1 reports the results on the sequence labeling task.[8] On the test set, Open-SESAME is the best model for both tasks. While best performance is not the core goal of this work, the fact that our best models perform in a similar range shows that our setup is sound to serve as a tool for comparing different pipeline variations.

Comparing our own models, we see that frame prediction performance is similar across setups: except for MULTILABEL, all F1-scores are within 3 points of each other. On role prediction, the setups that use MULTILABEL outperform the others. *Neo-traditional* performs the best on roles overall, whereas MULTITASK scores the worst. For frame prediction, performance does not seem to be boosted by joint role prediction. In fact, in MULTILABEL, performance on frames is very poor.

Similarly, adding roles as input for frame predictions (as in *Reverse-traditional*) does not help performance. Additional experiments to test the theoretical effectiveness of this strategy, using gold role labels as input, showed a slight improvement over FRAMESONLY (increasing F1 to 0.58 on test). However, when using predicted roles, we find no improvement and even see a small detrimental effect due to the poor performance of ROLESONLY. By contrast, *Neo-traditional* and *Merged*, when combining FRAMESONLY and MULTILABEL, perform well on both frames and roles. Lastly, MULTITASK does well on frames (but only slightly better than FRAMESONLY), but very poorly on roles.

Structural evaluation SemEval'07 scores are shown in Table 2. Note that two separate scores are reported for Open-SESAME: "true" and "recovered". For "true", we converted Open-SESAME predictions to SemEval format using all available structural information (i.e., links between roles, frames, and predicates); for "recovered", we first removed structural information and then attempted to recover it using our algorithm (see §5.2). The small difference between these scores suggests that recovery usually succeeds.

In any case, Open-SESAME consistently outperforms our models, and the difference is, overall, larger on the SemEval task than on the sequence labeling task. On the test set, *Merged* is our best model and has an F1-score within 0.05 of Open-SESAME using strict evaluation, and within 0.03 using partial span scoring. Interestingly, whereas the sequence-labeling performance of all models drops dramatically on the test set compared to the development set, SemEval task scores are more sta-

[7]Note that the Open-SESAME paper only treats `argID`, but models published at `https://github.com/swabhs/open-sesame` use the same architecture for doing `targetID` and `frameID` and are discussed at `https://github.com/swabhs/coling18tutorial`.

[8]For checking stability, all experiments were repeated three times and the scores averaged across runs. Overall, the models were quite stable and have F1-scores with standard deviations of ≤ 0.03. See the Appendix for full stability scores.

	Stripped						MULTILABEL					
	frames			roles			frames			roles		
	R	P	F	R	P	F	R	P	F	R	P	F
L02	0.65	0.72	0.68	0.36	0.43	0.39	0.55	0.64	0.59	0.34	0.45	0.39
L04	0.66	0.75	0.70	0.36	0.52	0.43	0.60	0.63	0.62	0.38	0.46	0.42
L06	0.62	0.77	0.69	0.39	0.47	0.42	0.59	0.67	0.62	0.39	0.58	0.47
L08	0.68	0.73	0.71	0.42	0.55	0.47	0.56	0.66	0.61	0.42	0.54	0.47
L10	0.71	0.72	0.71	0.45	0.52	0.48	0.56	0.62	0.59	0.47	0.55	0.51
L12	0.68	0.74	0.71	0.46	0.56	0.51	0.58	0.70	0.63	0.46	0.60	0.52
Mix	0.65	0.75	0.72	0.32	0.46	0.38	0.58	0.65	0.61	0.39	0.48	0.43

Table 3: Sequence-label scores (DEV) by BERT layer

ble. Finally, as expected, both Open-SESAME and our models get higher scores when partial credit is given to incomplete role spans, but our models benefit more from this than Open-SESAME does.

Out of our own models, *Merged* clearly wins, with a five points' difference to MULTILABEL, the worst-scoring model. A possible explanation for this difference is that MULTILABEL has poor recall for frame prediction: since frame structures always need to have a frame target, missing many frames is likely to cause low SemEval scores. However, good frames are not enough: while *Merged* beats Open-SESAME on frames on the development set, it has lower SemEval scores. More generally, it is interesting to note that good sequence labeling scores do not guarantee good SemEval performance. On one hand, we find that *Reverse-traditional* has good SemEval scores, especially for precision, even though it has poor sequence labeling scores on roles. On the other hand, *Neo-traditional* has good sequence labeling scores, but disappointing SemEval scores.

Consistency A factor that would be expected to lead to better SemEval scores is consistency between role and frame prediction: predicting many correct frames, but also many roles inconsistent with these frames, might lead to overall worse structures. Table 4 gives consistency scores (see §5.1) for all setups except *Stripped*. Open-SESAME and *Neo-traditional* score perfectly because frames are known at role prediction time, so that inconsistent roles are filtered out. There are large differences between the other setups: *Merged* has nearly 80% 'legal' roles, whereas *Joint*(MULTITASK) scores only 62%. Moreover, *Merged* outperforms MULTILABEL, despite getting its roles from MULTILABEL. We speculate that this is caused by MULTILABEL predicting 'orphaned' roles (i.e., correct roles lacking a matching frame) that are 're-parented' in *Merged*, which adds 'extra' frames from FRAMESONLY. Finally, *Reverse-traditional*'s consistency is lower than would be expected given that frame prediction is constrained by information about roles, which we attribute to poor role prediction in ROLESONLY. Still, *Reverse-traditional* performs quite well on SemEval, meaning that role coherence alone does not predict structural quality.

BERT layer analysis Analyzing the contributions of different BERT layers helps us better understand the implicit 'pipeline' learned by the model. Table 3 shows sequence labeling scores for the *Stripped* and MULTILABEL models, retrained using embeddings from individual layers. For comparison, the last row shows scores from Table 1.

We see an interesting discrepancy between frames and roles: role prediction clearly improves when using higher layers, but frame prediction is mostly stable, suggesting that the latter benefits from lexical information more than the former. This is true for both the Stripped and MULTILABEL models. Another interesting pattern is that role prediction is better for individual layers than for the "ScalarMix" setup, whereas this is not the case for frame prediction. This means that it is difficult to learn automatically which layers to use for role prediction, but it is yet unclear why.

7 Conclusions

We examined the frame-semantic parsing pipeline theoretically and practically, identifying 'chicken-or-egg' issues in the dependencies between subtasks, and studying them empirically within a BERT-based sequence-labeling framework.

We found that joint frame and role prediction works well, but not always better than using frames as input to roles. By contrast, previous studies (Yang and Mitchell, 2017; Peng et al., 2018) found substantial improvements from joint prediction. However, these systems use gold targets as input,

Experiment	**DEV**		**TEST**	
	score	**stdev**	**score**	**stdev**
Open-SESAME	*1.00*	*0.00*	*1.00*	*0.00*
Joint(MULTILABEL)	0.77	0.01	0.74	0.02
Joint(MULTITASK)	0.62	0.05	0.62	0.06
Neo-traditional(MULTILABEL)*†	1.00	0.00	1.00	0.00
Reverse-traditional(ROLESONLY)*†	0.71	0.06	0.70	0.04
Merged(MULTILABEL)*	0.80	0.01	0.78	0.02

Table 4: Consistency scores and deviance across runs

differ in architecture, and (partially) use different datasets, making direct comparison hard.

The main advantage of our sequence-labeling setup is the possibility to investigate frame and role prediction independently, as well as their mutual dependency. We found substantial benefits for role prediction from access to frame information through joint prediction or by receiving frames as input. For frame prediction, instead, the picture is less clear: while we found a theoretical benefit of using (gold) roles as input, this benefit disappears when using predicted roles. Similarly, when jointly predicting frames and roles, the MULTITASK setup yielded a slight improvement for frame prediction, whereas MULTILABEL deteriorated it. These results can be taken as supporting the traditional pipeline approach, but our results using SemEval evaluation, which looks at full frame structures, do not unequivocally confirm this: Open-SESAME performs best, but amongst our models, *Reverse-traditional* and *Merged* outperform the others, including *Neo-traditional*. This suggests that there might be valid alternatives to the standard pipeline, and exploring these might lead to a deeper understanding of frame semantic parsing task itself.

Our setup also allows for investigating which BERT layers both components use. Role prediction strongly prefers high BERT layers, while frame prediction is less picky, suggesting that the tasks use different linguistic information.

We see several logical extensions of our work. First, qualitative analysis of the overlaps in predictions from different models could shed light on the discrepancies between sequence labeling scores, consistency scores, and SemEval scores. A second direction would be to explore how our observations about the relationship between different components of the frame semantic parsing pipeline and BERT layers could be used to improve models. Finally, one could try more sophisticated architectures for sequence-labeling models, in particular by enforcing frame-role consistency within the model itself rather than during post-processing.

Acknowledgements

The research reported in this article was funded by the Dutch National Science organisation (NWO) through the project *Framing situations in the Dutch language*, `VC.GW17.083/6215`. We would also like to thank our anonymous reviewers for their helpful comments.

References

Waad Alhoshan, Riza Batista-Navarro, and Liping Zhao. 2019. Semantic frame embeddings for detecting relations between software requirements. In *Proceedings of the 13th International Conference on Computational Semantics - Student Papers*, pages 44–51, Gothenburg, Sweden. Association for Computational Linguistics.

Collin Baker, Michael Ellsworth, and Katrin Erk. 2007. SemEval-2007 task 19: Frame semantic structure extraction. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 99–104, Prague, Czech Republic. Association for Computational Linguistics.

Collin F. Baker, Charles J. Fillmore, and Beau Cronin. 2003. The structure of the FrameNet database. *International Journal of Lexicography*, 16(3):281–296.

Dipanjan Das. 2014. Statistical models for frame-semantic parsing. In *Proceedings of Frame Semantics in NLP: A Workshop in Honor of Chuck Fillmore (1929-2014)*, pages 26–29, Baltimore, MD, USA. Association for Computational Linguistics.

Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

C. Fillmore. 2006. Frame semantics. In D. Geeraerts, editor, *Cognitive Linguistics: Basic Readings*, pages 373–400. De Gruyter Mouton, Berlin, Boston. Originally published in 1982.

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 326–334, Boulder, Colorado. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Silvana Hartmann, Ilia Kuznetsov, Teresa Martin, and Iryna Gurevych. 2017. Out-of-domain FrameNet semantic role labeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 471–482, Valencia, Spain. Association for Computational Linguistics.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 364–369, Melbourne, Australia. Association for Computational Linguistics.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Alexandre Kabbach, Corentin Ribeyre, and Aurélie Herbelot. 2018. Butterfly effects in frame semantic parsing: impact of data processing on model ranking. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3158–3169, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A. Smith. 2018. Learning joint semantic parsers from disjoint data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1492–1502, New Orleans, Louisiana. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Drew Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme. 2015. Semantic proto-roles. *Transactions of the Association for Computational Linguistics*, 3:475–488.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. 2017. Frame-semantic parsing with softmax-margin segmental rnns and a syntactic scaffold. *CoRR*, abs/1706.09528.

Songbo Tan. 2007. Using error-correcting output codes with model-refinement to boost centroid text classifier. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 81–84, Prague, Czech Republic. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Kristina Toutanova, Aria Haghighi, and Christopher Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 589–596, Ann Arbor, Michigan. Association for Computational Linguistics.

Bishan Yang and Tom Mitchell. 2017. A joint sequential and relational model for frame-semantic parsing.

In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, Copenhagen, Denmark. Association for Computational Linguistics.

A Supplemental material

A.1 Model stability

In order to check for model stability, we repeated all our experiments (excl. the experiments for BERT layer analysis) three times. The standard deviations of the sequence-labeling scores are shown in Table A.1. F1-scores seem quite stable overall, but in a few cases there are larger deviations in precision and/or recall, especially in the *Joint*(MULTITASK) model.

	frames			roles		
Experiment	**R**	**P**	**F**	**R**	**P**	**F**
Joint(MULTILABEL)	01	00	00	00	01	01
Joint(MULTITASK)	02	02	00	04	07	01
Neotrad.(MULTILABEL, Sp.)	02	01	01	02	04	01
Merged(MULTILABEL)	01	01	00	00	01	01
Stripped(FRAMESONLY, Emb.)	02	04	01	-	-	-
Stripped(FRAMESONLY, Sp.)	03	02	01	-	-	-
Stripped(ROLESONLY)	-	-	-	02	04	03

Table A.1: Model stability: standard deviation (%) across runs of sequence-labeling scores (on DEV)

Split	#Sentences	#Frame structures
`train`	3,413	19,391
`dev`	387	2,272
`test`	2,420	6,714

Table A.2: FrameNet corpus stats

A.2 FrameNet data

Corpus We used the standard FrameNet corpus (release 1.7) for all experiments. We used the `fulltext.train` split for training, the `dev` split for validation and evaluation, and the `test` split for final evaluation. Table A.2 shows the relative sizes of these splits.

Distribution of roles One of the key simplifications of our sequence labeling setup is 'decoupling' frames and roles. This reduces the label space since some roles occur in many different frames. Figure A.1 shows the most frequent role names with the number of different frames that they occur in. As can be seen from the graph, most frequent roles are very general ones such as 'Time', 'Place', 'Manner', etc. Although roles are, in the FrameNet philosophy, strictly defined relative to frames, we expect that roles sharing a name across frames will have a very similar semantics.

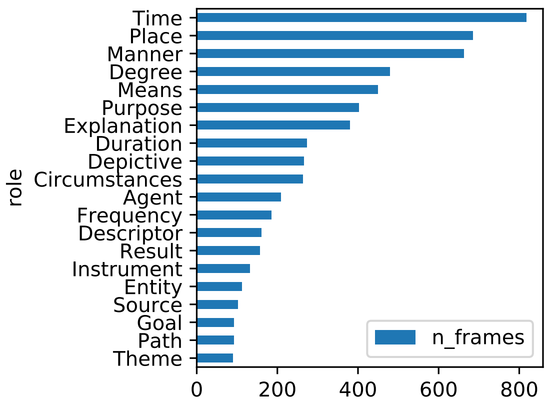

Figure A.1: Top-20 role names by number of frames