SPECTRA: Sparse Structured Text Rationalization

Nuno M. Guerreiro[1,2] and **André F. T. Martins**[1,2,3]
[1]Instituto de Telecomunicações, Lisbon, Portugal
[2]Instituto Superior Técnico & LUMLIS (Lisbon ELLIS Unit), Lisbon, Portugal
[3]Unbabel, Lisbon, Portugal
{nuno.s.guerreiro, andre.t.martins}@tecnico.ulisboa.pt

Abstract

Selective rationalization aims to produce decisions along with rationales (e.g., text highlights or word alignments between two sentences). Commonly, rationales are modeled as stochastic binary masks, requiring sampling-based gradient estimators, which complicates training and requires careful hyperparameter tuning. Sparse attention mechanisms are a deterministic alternative, but they lack a way to regularize the rationale extraction (e.g., to control the sparsity of a text highlight or the number of alignments). In this paper, we present a unified framework for deterministic extraction of structured explanations via constrained inference on a factor graph, forming a differentiable layer. Our approach greatly eases training and rationale regularization, generally outperforming previous work on what comes to performance and plausibility of the extracted rationales. We further provide a comparative study of stochastic and deterministic methods for rationale extraction for classification and natural language inference tasks, jointly assessing their predictive power, quality of the explanations, and model variability.

1 Introduction

Selective rationalization (Lei et al., 2016; Bastings et al., 2019; Swanson et al., 2020) is a powerful explainability method, in which we construct models (*rationalizers*) that produce an explanation or *rationale* (e.g: text highlights or alignments; Zaidan et al. 2007) along with the decision.

One, if not the main, drawback of rationalizers is that it is difficult to train the generator and the predictor jointly under instance-level supervision (Jain et al., 2020). Hard attention mechanisms that stochastically sample rationales employ regularization to encourage sparsity and contiguity, and make it necessary to estimate gradients using the score function estimator (SFE), also known as REINFORCE (Williams, 1992), or reparameterized gradients (Kingma and Welling, 2014; Jang et al., 2017). Both of these factors substantially complicate training by requiring sophisticated hyperparameter tuning and lead to brittle and fragile models that exhibit high variance over multiple runs. Other works use strategies such as top-k to map token-level scores to rationales, but also require gradient estimations to train both modules jointly (Paranjape et al., 2020; Chang et al., 2020). In turn, sparse attention mechanisms (Treviso and Martins, 2020) are deterministic and have exact gradients, but lack a direct way to control sparsity and contiguity in the rationale extraction. This raises the question: *how can we build an easy-to-train fully differentiable rationalizer that allows for flexible constrained rationale extraction?*

To answer this question, we introduce **sparse structured text rationalization** (**SPECTRA**), which employs LP-SparseMAP (Niculae and Martins, 2020), a constrained structured prediction algorithm, to provide a deterministic, flexible and modular rationale extraction process. We exploit our method's inherent flexibility to extract highlights and interpretable text matchings with a diverse set of constraints.

Our contributions are:

- We present a unified framework for deterministic extraction of structured rationales (§3) such as constrained highlights and matchings;
- We show how to add constraints on the rationale extraction, and experiment with several structured and hard constraint factors, exhibiting the modularity of our strategy;
- We conduct a rigorous comparison between deterministic and stochastic rationalizers (§4) for both highlights and matchings extraction.

Experiments on selective rationalization for sentiment classification and natural language inference (NLI) tasks show that our proposed approach

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 6534–6550
November 7–11, 2021.

Method	Deterministic Training	Exact Gradients	Constrained Extraction	Encourages Contiguity
Lei et al. (2016)	✗	✗	✗	✓
Bastings et al. (2019)	✗	✗	✓	✓
Treviso and Martins (2020)	✓	✓	✗	✗
SPECTRA (ours)	✓	✓	✓	✓

Table 1: Positioning of our approach in the literature of rationalization for highlights extraction. Our method is an easy-to-train fully differentiable deterministic rationalizer that allows for flexible rationale regularization.

achieves better or competitive performance and similarity with human rationales, while exhibiting less variability and easing rationale regularization when compared to previous approaches.[1]

2 Background

2.1 Rationalization for Highlights Extraction

Rationalization models for highlights extraction, also known as *select-predict* or *explain-predict* models (Jacovi and Goldberg, 2021; Zhang et al., 2021b), are based on a cooperative framework between a rationale generator and a predictor: the generator component encodes the input text and extracts a "rationale" (e.g., a subset of highlighted words), and the predictor classifies the input conditioned only on the extracted rationale. Typically, this is done by obfuscating the words that are not in the rationale with a binary mask.

Highlights Extraction. We consider a standard text classification or regression setup, in which we are given an input sequence $\boldsymbol{x} \in \mathbb{R}^{D \times L}$, where D is the embedding size and L is the sequence length (number of words), and we want to predict its corresponding label $y \in \mathbb{R}$ for regression or $y \in \{1, \ldots, C\}$ for classification. A generator model, gen, encodes the input text $\boldsymbol{x}$ into token-level scores. Then, a rationale $\boldsymbol{z}$, e.g. a binary mask over the tokens, is extracted based on these scores. Subsequently, the predictor model makes predictions conditioned only on the rationale $\hat{y} = \mathsf{pred}(\boldsymbol{z} \odot \boldsymbol{x})$, where $\odot$ denotes the Hadamard (elementwise) product.

End-to-end Training and Testing Procedure. While most rationalization methods deterministically select the rationale at test time, there are differences on how these models are **trained**. For instance, Lei et al. (2016) and Bastings et al. (2019) use stochastic binary variables (Bernoulli and HardKuma, respectively), and sample the rationale $\boldsymbol{z} \sim \mathsf{gen}(\boldsymbol{x}) \in \{0, 1\}^L$, whereas Treviso and Martins (2020) make a continuous relaxation of these binary variables and define the rationale as a sparse probability distribution over the tokens, $\boldsymbol{z} = \mathsf{sparsemax}(\mathsf{gen}(\boldsymbol{x}))$ or $\boldsymbol{z} = \alpha\text{-}\mathsf{entmax}(\mathsf{gen}(\boldsymbol{x}))$. In the latter approach, instead of a binary vector, we have $\boldsymbol{z} \in \triangle^{L-1}$, where $\triangle^{L-1}$ is the $L-1$ probability simplex $\triangle^{L-1} := \{\boldsymbol{p} \in \mathbb{R}^L : \mathbf{1}^\top \boldsymbol{p} = 1, \boldsymbol{p} \geq 0\}$. Words receiving non-zero probability are considered part of the rationale.

Rationalizers that use hard attention mechanisms or heuristics to extract the rationales are distinctively hard to train end-to-end, as they require marginalization over all possible rationales, which is intractable in practice. Thus, recourse to sampling-based gradient estimations is a necessity, either via REINFORCE-style training, which exhibits high variance (Lei et al., 2016; Chang et al., 2020), or via reparameterized gradients (Bastings et al., 2019; Paranjape et al., 2020). This renders training these models a complex and cumbersome task. These approaches are often brittle and fragile for the high sensitivity that they show to changes in the hyperparameters and to variability due to sampling. On the other hand, existing rationalizers that use sparse attention mechanisms (Treviso and Martins, 2020) such as sparsemax attention, while being deterministic and end-to-end differentiable, do not have a direct handle to constrain the rationale in terms of sparsity and contiguity. We endow them with these capabilities in this paper as shown in Table 1, where we position our work in the literature for highlights extraction.

Constrained Rationale Extraction. Existing rationalizers are *extractive*: they select and extract words or word pairs to form the rationale. Since a rationalizer that extracts the whole input would be meaningless as an explainer, they must have a length constraint or a sparsity inducing component. Moreover, rationales are idealized to encourage selection of contiguous words, as there is some

[1] Our library for rationalization is available at https://github.com/deep-spin/spectra-rationalization.

evidence that this improves readibility (Jain et al., 2020). Some works opt to introduce regularization terms placed on the binary mask such as the ℓ_1 norm and the fused-lasso penalty to encourage sparse and compact rationales (Lei et al., 2016; Bastings et al., 2019). Others use hard constraints through heuristics such as top-k, which is not contiguous but sparse, or select a chunk of text with a pre-specified length that corresponds to the highest total score over all possible spans of that length (Chang et al., 2020; Paranjape et al., 2020; Jain et al., 2020). Sparse attention mechanisms can also be used to extract rationales, but since the rationales are constrained to be in the simplex, controlling the number of selected tokens and simultaneously promoting contiguity is non-trivial.

2.2 Rationalization for Matchings Extraction

For this task, we consider a natural language inference setup in which classification is made based on two input sentences: a premise $\boldsymbol{x}_P \in \mathbb{R}^{D \times L_P}$ and a hypothesis $\boldsymbol{x}_H \in \mathbb{R}^{D \times L_H}$, where L_P and L_H are the sequence lengths of the premise and hypothesis, respectively, and D is the embedding size. A generator model (gen) encodes $\boldsymbol{x}_P$ and $\boldsymbol{x}_H$ separately and then computes pairwise costs between the encoded representations to produce a score matrix $\boldsymbol{S} \in \mathbb{R}^{L_P \times L_H}$. The score matrix $\boldsymbol{S}$ is then used to compute an alignment matrix $\boldsymbol{Z} \in \mathbb{R}^{L_P \times L_H}$, where $z_{ij} = 1$ if the i^{th} premise word is aligned to the j^{th} word in the hypothesis. $\boldsymbol{Z}$ subsequently acts as a sparse mask to obtain text representations that are aggregated with the original encoded sequences and fed to a predictor to obtain the output predictions.

2.3 Structured Prediction on Factor Graphs

Finding the highest scored rationale under the constraints described above is a structured prediction problem, which involves searching over a very large and combinatorial space. We assume that a rationale $\boldsymbol{z}$ can be represented as an L-dimensional binary vector. For example, in highlights extraction, L is the number of words in the document and $\boldsymbol{z}$ is a binary mask selecting the relevant words; and in the extraction of matchings, $L = L_P \times L_H$ and $\boldsymbol{z}$ is a flattened binary vector whose entries indicate if a premise word is aligned to a word in the hypothesis. We let $\mathcal{Z} \subseteq \{0,1\}^L$ be the set of rationales that satisfy the given constraints, and let $\boldsymbol{s} = \mathsf{gen}(\boldsymbol{x}) \in \mathbb{R}^L$ be a vector of scores.

Factor Graph. In the sequel, we consider problems that consist of multiple interacting subproblems. Niculae and Martins (2020) present structured differentiable layers, which decompose a given problem into simpler subproblems, instantiated as local factors that must agree when overlapped. Formally, we assume a factor graph $\mathcal{F}$, where each factor $f \in \mathcal{F}$ corresponds to a subset of variables. We denote by $\boldsymbol{z}_f = (\boldsymbol{z}_i)_{i \in f}$ the vector of variables corresponding to factor f. Each factor has a local score function $h_f(\boldsymbol{z}_f)$. Examples are **hard constraint factors**, which take the form

$$h_f(\boldsymbol{z}_f) = \begin{cases} 0 & \text{if } \boldsymbol{z}_f \in \mathcal{Z}_f \\ -\infty & \text{otherwise,} \end{cases} \tag{1}$$

where $\mathcal{Z}_f$ is a polyhedral set imposing hard constraints (see Table 2 for examples); and **structured factors**, which define more complex functions with structural dependencies on $\boldsymbol{z}_f$, such as

$$h_f(\boldsymbol{z}_f) = \sum_{i=1}^{L-1} r_{i,i+1} z_{i,i+1}, \tag{2}$$

where $r_{i,i+1} \in \mathbb{R}$ are edge scores, which together define a **sequential factor**. We require that for any factor the following local subproblem is tractable:

$$\hat{\boldsymbol{z}}_f = \arg \max_{\boldsymbol{z}_f \in \{0,1\}^{|f|}} \boldsymbol{s}_f^\top \boldsymbol{z}_f + h_f(\boldsymbol{z}_f). \tag{3}$$

MAP inference. The problem of identifying the highest-scoring global structure, known as **maximum *a posteriori*** (MAP) **inference**, is written as:

$$\hat{\boldsymbol{z}} = \arg \max_{\boldsymbol{z} \in \{0,1\}^L} \underbrace{\left(\boldsymbol{s}^\top \boldsymbol{z} + \sum_{f \in \mathcal{F}} h_f(\boldsymbol{z}_f)\right)}_{\text{score}(\boldsymbol{z}; \boldsymbol{s})}. \tag{4}$$

The objective being maximized is the global score function $\text{score}(\boldsymbol{z}; \boldsymbol{s})$, which combines information coming from all factors. The solution of the MAP problem is a vector $\hat{\boldsymbol{z}}$ whose entries are zeros and ones. However, it is often difficult to obtain an exact maximization algorithm for complex structured problems that involve interacting subproblems that impose global agreement constraints.

Gibbs distribution and sampling. The global score function can be used to define a Gibbs distribution $p(\boldsymbol{z}; \boldsymbol{s}) \propto \exp(\text{score}(\boldsymbol{z}; \boldsymbol{s}))$. The MAP in (4) is the mode of this distribution. Sometimes (e.g. in stochastic rationalizers) we want to sample from this distribution, $\hat{\boldsymbol{z}} \sim p(\boldsymbol{z}; \boldsymbol{s})$. Exact,

unbiased samples are often intractable to obtain, and approximate sampling strategies have to be used, such as perturb-and-MAP (Papandreou and Yuille, 2011; Corro and Titov, 2019a,b). These strategies necessitate gradient estimators for end-to-end training, which are often obtained via REINFORCE (Williams, 1992) or reparametrized gradients (Kingma and Welling, 2014; Jang et al., 2017).

LP-MAP inference. In many cases, the MAP problem (4) is intractable due to the overlapping interaction of the factors $f \in \mathcal{F}$. A commonly used relaxation is to replace the integer constraints $\boldsymbol{z} \in \{0,1\}^L$ by continuous constraints, leading to:

$$\hat{\boldsymbol{z}} = \arg \max_{\boldsymbol{z} \in [0,1]^L} \text{score}(\boldsymbol{z}; \boldsymbol{s}). \quad (5)$$

The problem above is known as LP-MAP inference (Wainwright and Jordan, 2008). In some cases (for example, when the factor graph $\mathcal{F}$ does not have cycles), LP-MAP inference is *exact*, i.e., it gives the same results as MAP inference. In general, this does not happen, but for many problems in NLP, LP-MAP relaxations are often nearly optimal (Koo et al., 2010; Martins et al., 2015). Importantly, computation in the hidden layer of these problems may render the network unsuitable for gradient-based training, as with MAP inference.

LP-SparseMAP inference. The optimization problem respective to LP-SparseMAP is the ℓ_2 regularized LP-MAP (Niculae and Martins, 2020):

$$\hat{\boldsymbol{z}} = \arg \max_{\boldsymbol{z} \in [0,1]^L} \left(\text{score}(\boldsymbol{z}; \boldsymbol{s}) - 1/2 \|\boldsymbol{z}\|^2\right). \quad (6)$$

Unlike MAP and LP-MAP, the LP-SparseMAP relaxation is suitable to train with gradient back-propagation. Moreover, it favors sparse vectors $\hat{\boldsymbol{z}}$, i.e., vectors that have only a few non-zero entries. One of the most appealing features of this method is that it is modular: an arbitrary complex factor graph can be instantiated as long as a MAP oracle for each of the constituting factors is provided. This approach generalizes SparseMAP (Niculae et al., 2018), which requires an exact MAP oracle for the factor graph in its entirety. In fact, LP-SparseMAP recovers SparseMAP when there is a single factor $\mathcal{F} = \{f\}$. By only requiring a MAP oracle for each $f \in \mathcal{F}$, LP-SparseMAP makes it possible to instantiate more expressive factor graphs for which MAP is typically intractable. Table 2 lists several logic constraint factors which are used in this paper.

Factor Name	Imposed Constraint
XOR	$\sum z_k = 1$
AtMostOne	$\sum z_k \leq 1$
BUDGET	$\sum z_k \leq B$

Table 2: Collection of logic factors and its imposed constraints. Each of these factors defines a constraint set $\mathcal{Z}_f$ as described in §2.3.

3 Deterministic Structured Rationalizers

The idea behind our approach for selective rationalization is very simple: leverage the inherent flexibility and modularity of LP-SparseMAP for constrained, deterministic and fully differentiable rationale extraction.

3.1 Highlights Extraction

Model Architecture. We use the model setting described in §2. First, a generator model produces token-level scores $s_i, i \in \{1, \ldots, L\}$. We propose replacing the current rationale extraction mechanisms (e.g. sampling from a Bernoulli distribution, or using sparse attention mechanisms) with an LP-SparseMAP extraction layer that computes token-level values $\hat{\boldsymbol{z}} \in [0,1]^L$, which are then used to mask the original sequence for prediction. Due to LP-SparseMAP's propensity for sparsity, many entries in $\hat{\boldsymbol{z}}$ will be zero, which approaches what is expected from a binary mask.

Factor Graphs. The definition of the factor graph $\mathcal{F}$ is central to the rationale extraction, as each of the local factors $f \in \mathcal{F}$ will impose constraints on the highlight. We start by instantiating a factor graph with L binary variables (one for each token) and a pairwise factor for every pair of contiguous tokens:

$$\mathcal{F} = \{\textsf{PAIR}(z_i, z_{i+1}; r_{i,i+1}) : 1 \leq i < L\}, \quad (7)$$

which yields the binary pairwise MRF (§2.3)

$$\text{score}(\boldsymbol{z}; \boldsymbol{s}) = \sum_{i=1}^{L} s_i z_i + \sum_{i=1}^{L-1} r_{i,i+1} z_i z_{i+1}. \quad (8)$$

Instantiating this factor with non-negative edge scores, $r_{i,i+1} \geq 0$, encourages contiguity on the rationale extraction. Making use of the modularity of the method, we impose sparsity by further adding a BUDGET factor (see Table 2):

$$\begin{aligned}\mathcal{F} = &\{\textsf{PAIR}(z_i, z_{i+1}; r_{i,i+1}) : 1 \leq i < L\} \\ &\cup \{\textsf{BUDGET}(z_1, \ldots, z_L; B)\}. \quad (9)\end{aligned}$$

The size of the rationale is constrained to be, at most, $B\%$ of the input document size. Intuitively, the lower the B, the shorter the extracted rationales will be. Notice that this graph is composed of L local factors. Thus, LP-SparseMAP would have to enforce agreement between all these factors in order to compute $\boldsymbol{z}$. Interestingly, factor graph representations are usually not unique. In our work, we instantiate an equivalent formulation of the factor graph in Eq. 9 that consists of a single factor, **H:SeqBudget**. This factor can be seen as an extension of that of the LP-Sequence model in Niculae and Martins (2020): a linear-chain Markov factor with MAP provided by the Viterbi algorithm (Viterbi, 1967; Rabiner, 1989). The difference resides in the additional budget constraints that are incorporated in the MAP decoding. These constraints can be handled by augmenting the number of states in the dynamic program to incorporate how many words in the budget have already been consumed at each time step, leading to time complexity $\mathcal{O}(LB)$.

3.2 Matchings Extraction

Model Architecture. Our architecture is inspired by ESIM (Chen et al., 2017). First, a generator model encodes two documents $\boldsymbol{x}_P$, $\boldsymbol{x}_H$ separately to obtain the encodings $(\tilde{\boldsymbol{h}}_1^P, \ldots, \tilde{\boldsymbol{h}}_{L_P}^P)$ and $(\tilde{\boldsymbol{h}}_1^H, \ldots, \tilde{\boldsymbol{h}}_{L_H}^H)$, respectively. Then, we compute alignment dot-product pairwise scores between the encoded representations to produce a score matrix $\boldsymbol{S} \in \mathbb{R}^{L_P \times L_H}$ such that $s_{ij} = \langle \tilde{\boldsymbol{h}}_i^P, \tilde{\boldsymbol{h}}_j^H \rangle$. We use LP-SparseMAP to obtain $\boldsymbol{Z}$, a constrained structured symmetrical alignment $\boldsymbol{Z}$ in which $z_{ij} \in [0, 1]$, as described later. Then, we "augment" each word in the premise and hypothesis with the corresponding aligned weighted average by computing $\bar{\boldsymbol{h}}_i^P = \left[\tilde{\boldsymbol{h}}_i^P, \sum_j z_{ij}\, \tilde{\boldsymbol{h}}_j^H\right]$ and $\bar{\boldsymbol{h}}_j^H = \left[\tilde{\boldsymbol{h}}_j^H, \sum_i z_{ji}\, \tilde{\boldsymbol{h}}_i^P\right]$, and separately feed these vectors to another encoder and pool to find representations $\boldsymbol{r}^P$ and $\boldsymbol{r}^H$. Finally, the feature vector $\boldsymbol{r} = [\boldsymbol{r}^P, \boldsymbol{r}^H, \boldsymbol{r}^P - \boldsymbol{r}^H, \boldsymbol{r}^P \odot \boldsymbol{r}^H]$ is fed to a classification head for the final prediction. We also experiment with a strategy in which we assume that the hypothesis is known and the premise is masked for *faithful* prediction. We consider $\bar{\boldsymbol{h}}_i^P = \left[\sum_j z_{ij}\, \tilde{\boldsymbol{h}}_j^H\right]$, such that the only information about the premise that the model has to make a prediction comes from the alignment and its masking of the encoded representation.

Factor Graphs. We instantiate three different factor graphs for matchings extraction. The first – **M:XorAtMostOne** – is the same as the LP-Matching factor used in Niculae and Martins (2020) with one XOR factor per row and one AtMostOne factor per column:

$$\mathcal{F} = \{\mathsf{XOR}(z_{i1}, ..., z_{in}) : 1 \leq i \leq L_P \quad (10)$$
$$\cup\, \{\mathsf{AtMostOne}(z_{1j}, ..., z_{mj}) : 1 \leq j \leq L_H\}$$

which requires at least one active alignment for each word of the premise, since the i^{th} word in the premise **must** be connected to the hypothesis. The j^{th} word in the hypothesis, however, is not constrained to be aligned to any word in the premise. In the second factor graph – **M:AtMostOne2** – we alleviate the XOR restriction on the premise words to an AtMostOne restriction. The expected output is a sparser matching for there is no requirement of an active alignment for each word of the premise. The third factor graph – **M:Budget** – allows us to have more refined control on the sparsity of the resulting matching, by adding an extra global BUDGET factor (with budget B) to the factor graph of M:AtMostOne2 so that the resulting matching will have at most B active alignments.

Stochastic Matchings Extraction. Prior work for selective rationalization of text matching uses constrained variants of optimal transport to obtain the rationale (Swanson et al., 2020). Their model is end-to-end differentiable using the Sinkhorn algorithm (Cuturi, 2013a). Thus, in order to provide a comparative study of stochastic and deterministic methods for rationalization of text matchings, we implement a perturb-and-MAP rationalizer (§2.3). We perturb the scores s_{ij} by computing $\tilde{\boldsymbol{S}} = \boldsymbol{S} + \boldsymbol{P}$, in which each element of $\boldsymbol{P}$ contains random samples from the Gumbel distribution, $p_{ij} \sim \mathcal{G}(0, 1)$. We utilize these perturbed scores to compute non-symmetrical alignments from the premise to the hypothesis and vice-versa, such that their entries are in $[0, 1]$. At test time, we obtain the most probable matchings, such that their entries are in $\{0, 1\}$. These matchings are such that every word in the premise **must** be connected to a single word in the hypothesis and vice-versa.

4 Experimental Setup

4.1 Highlights for Sentiment Classification

Data and Evaluation. We used the SST, AgNews, IMDB, and Hotels datasets for text clas-

sification and the BeerAdvocate dataset for regression. The statistics and details of all datasets can be found in §A. The rationale specified lengths, as percentage of each document, for the strategies that impose fixed sparsity are 20% for the SST, AgNews and IMDB datasets, 15% for the Hotels dataset, and 10% for the BeerAdvocate dataset. We evaluate end task performance (Macro F_1 for classification tasks and MSE for regression), and matching with human annotations through token-level F_1 score (DeYoung et al., 2019) for the datasets that contain human annotations.

Baselines. We compare our results with three versions of the stochastic rationalizer of Lei et al. (2016): the original one – **SFE** – which uses the score function estimator to estimate the gradients; a second one – **SFE w/ Baseline** – which uses SFE with a moving average baseline variance reduction technique; a third – **Gumbel** – in which we employ the Gumbel-Softmax reparameterization (Jang et al., 2017) to reparameterize the Bernoulli variables; and, a fourth – **HardKuma** – in which we employ HardKuma variables (Bastings et al., 2019) instead of Bernoulli variables and use reparameterized gradients for training end-to-end. Moreover, the latter rationalizer employs a Lagrangian relaxation to solve the constrained optimization problem of targeting specific sparsity rates. We also experimented with two deterministic strategies that use sparse attention mechanisms: a first that utilizes **sparsemax** (Martins and Astudillo, 2016), and a second that utilizes **fusedmax** (Niculae and Blondel, 2019) which encourages the network to pay attention to contiguous segments of text, by adding an additional total variation regularizer, inspired by the fused lasso. It is a natural deterministic counterpart of the constrained rationalizer proposed by Lei et al. (2016), since the regularization encourages both sparsity and contiguity. The use of fusedmax for this task is new to the best of our knowledge. Similarly to Jain et al. (2020), we found that the stochastic rationalizers of Lei et al. (2016) and its variants (SFE, SFE w/ Baseline and Gumbel) require cumbersome hyperparameter search and tend to degenerate in such a way that the generated rationales are either the whole input text or empty text. Thus, at inference time, we follow the strategy proposed by Jain et al. (2020) and restrict the generated rationale to a specified length ℓ via two mappings: **contiguous**, in which the span of length ℓ, out of all the spans of this length, whose token-level scores cumulative sum is the highest is selected; and **top-k**, in which the ℓ tokens with highest token-level scores are selected. Contrary to (Jain et al., 2020), for the rationalizer of Bastings et al. (2019) (HardKuma), we carefully tuned both the model hyperparameters and the Lagrangian relaxation algorithm hyperparameters, so as to use the deterministic policy in testing time that they propose.[2] All implementation details can be found in §C. We also report the full-text baselines for each dataset in §D.

4.2 Matchings for Natural Language Inference

Data and Evaluation. We used the English language SNLI and MNLI datasets (Bowman et al., 2015; Chen et al., 2017). We evaluate end task performance for both datasets. For the experiments with the M:Budget, we used a fixed budget of $B = 4$ for SNLI and $B = 6$ for MNLI. We also conduct further experiments with the HANS dataset (McCoy et al., 2019) which aims to analyse the use of linguistic heuristics (lexical overlap, constituent and subsequence heuristics) of NLI systems. The statistics and details of each dataset can be found in §B.

Baselines. We compare our results with variants of constrained optimal transport for selective rationalization employed by Swanson et al. (2020): relaxed 1:1, which is similar in nature to our proposed M:AtMostOne2 factor; and exact $k = 4$ similar to our proposed M:Budget with budget $B = 4$. We also replicate the LP-matching implementation of Niculae and Martins (2020) which consists of the original ESIM model described in §3.2 with $\boldsymbol{Z}$ as the output of the LP-SparseMAP problem with a M:XorAtMostOne factor. Importantly, both these models aggregate the encoded premise representation with the information that comes from the alignment. All implementation details can be found in §C. We also report the ESIM baselines in §D.

5 Results and Analysis

5.1 Extraction of Text Highlights

Predictive Performance. We report the predictive performances of all models in Table 3. We

[2] We have found that using the deterministic policy at testing time proposed by Bastings et al. (2019) instead of the top-k or contiguous strategies is critical to achieve good performance with the HardKuma rationalizer.

Method	Rationale	SST ↑	AgNews ↑	IMDB ↑	Beer ↓	Hotels ↑
🎲 SFE	top-k	.76 (.71/.80)	.92 (.92/.92)	.84 (.72/.88)	.018 (.016/.020)	.66 (.62/.69)
	contiguous	.71 (.68/.75)	.86 (.85/.86)	.65 (.57/.73)	.020 (.019/.024)	.62 (.34/.72)
🎲 SFE w/ Baseline	top-k	.78 (.76/.80)	.92 (.92/.93)	.82 (.72/.88)	.019 (.017/.020)	.56 (.34/.64)
	contiguous	.70 (.64/.75)	.86 (.84/.86)	.76 (.73/.80)	.021 (.019/.025)	.55 (.34/.69)
🎲 Gumbel	top-k	.70 (.67/.72)	.78 (.73/.84)	.74 (.71/.78)	.026 (.018/.041)	.83 (.73/.92)
	contiguous	.67 (.67/.68)	.77 (.74/.81)	.72 (.72/.73)	.043 (.040/.048)	.74 (.65/.84)
🎲 HardKuma	–	.80 (.80/.81)	.90 (.87/.88)	.87 (.90/.91)	.019 (.016/.020)	.90 (.88/.92)
🎯 Sparse Attention	sparsemax	**.82** (.81/.83)	**.93** (.93/.93)	.89 (.89/.90)	.019 (.016/.021)	.89 (.87/.92)
	fusedmax	.81 (.81/.82)	.92 (.91/.92)	.88 (.87/.89)	.018 (.017/.019)	.85 (.77/.90)
🎯 SPECTRA (ours)	H:SeqBudget	.80 (.79/.81)	.92 (.92/.93)	**.90** (.89/.90)	**.017** (.016/.019)	**.91** (.90/.92)

Table 3: Model predictive performances across datasets, for stochastic (🎲) and deterministic (🎯) methods. We report mean and min/max F_1 scores across five random seeds on test sets for all datasets but Beer where we report MSE. We bold the best-performing rationalized model(s) for each corpus.

Method	Rationale	SST	AgNews	IMDB	Beer	Hotels
🎲 HardKuma	–	.15 (.12/.19)	.19 (.18/.19)	.03 (.02/.03)	.08 (.00/.17)	.09 (.07/.12)
🎯 Sparse Attention	sparsemax	.17 (.13/.23)	.13 (.11/.15)	.02 (.02/.03)	.11 (.09/.13)	.03 (.02/.04)
	fusedmax	.60 (.14/1.0)	.32 (.10/.66)	.02 (.01/.02)	.26 (.03/.98)	.04 (.01/.08)

Table 4: Average size of the extracted rationales using the HardKuma stochastic rationalizer (🎲) and deterministic (🎯) sparse attention mechanisms. We report mean and min/max average size across five random seeds.

observe that the deterministic rationalizers that use sparse attention mechanisms generally outperform the stochastic rationalizers while exhibiting lower variability across different random seeds and different datasets. In general and as expected, for the stochastic models, the top-k strategy for rationale extraction outperforms the contiguous strategy. As reported in Jain et al. (2020), strategies that impose a contiguous mapping trade coherence for performance on the end-task. Our experiments also show that HardKuma is the stochastic rationalizer least prone to variability across different seeds, faring competitively with the deterministic methods. The strategy proposed in this paper, H:SeqBudget, fares competitively with the deterministic methods and generally outperforms the stochastic methods. Moreover, similarly to the other deterministic rationalizers, our method exhibits lower variability across different runs. We show examples of highlights extracted by SPECTRA in §G.

5.2 Quality of the Rationales

Rationale Regularization. We report in Table 4 the average size of the extracted rationales (proportion of words not zeroed out) across datasets for the stochastic HardKuma rationalizer and for each rationalizer that uses sparse attention mechanisms. The latter strategies do not have any mechanism to regularize the sparsity of the extracted rationales, which leads to variability on the rationale extraction. This is especially the case for the fusedmax strategy, as it pushes adjacent tokens to be given the same attention probability. This might lead to rationale degeneration when the attention weights are similar across all tokens. On the other hand, HardKuma employs a Lagrangian relaxation algorithm to target a predefined sparsity level. We have found that careful hyperparameter tuning is required across different datasets. While, generally, the average size of the extracted rationales does not exhibit considerable variability, some random seeds led to degeneration (the model extracts empty rationales). Remarkably, our proposed strategy utilizes the BUDGET factor to set a predefined desired rationale length, regularizing the rationale extraction while still applying a deterministic policy that exhibits low variability across different runs and datasets (Table 3).

Matching with Human Annotations. We report token-level F_1 scores in Table 5 to evaluate the quality of the rationales for the datasets for which we had human annotations for the test set. We observe that our proposed strategy and HardKuma outperform all the other methods on what concerns matching the human annotations. This was to be expected considering the results shown in Table 3 and Table 4: the stochastic models other than HardKuma do not fare competitively with the deterministic models and their variability across

Method	Rationale	Beer	Hotels
🎲 SFE	top-k	.19 (.13/.30)	.16 (.12/.30)
	contiguous	.35 (.18/.42)	.14 (.12/.15)
🎲 SFE w/ Baseline	top-k	.17 (.14/.19)	.14 (.13/.18)
	contiguous	.41 (.37/.42)	.15 (.14/.15)
🎲 Gumbel	top-k	.27 (.14/.39)	.36 (.27/.48)
	contiguous	.42 (.41/.42)	.36 (.29/.48)
🎲 HardKuma	–	.37 (.00/.90)	**.52** (.37/.57)
🎯 Sparse Attention	sparsemax	.48 (.41/.55)	.17 (.07/.31)
	fusedmax	.39 (.29/.53)	.25 (.09/.31)
🎯 SPECTRA (ours)	H:SeqBudget	**.61** (.56/.68)	.37 (.34/.40)

Table 5: Evaluation of the rationales through matching with human annotations, for stochastic (🎲) and deterministic (🎯) methods. We report mean token-level F_1 scores and min/max across five random seeds.

Matching Structure	SNLI	MNLI
Not Faithful		
🎯 OT relaxed 1:1†	.82	–
🎯 OT exact $k = 4$†	.81	–
🎲 Gumbel Matching	.85 (.84/.85)	.73 (.72/.73)
🎯 M:XorAtMostOne	**.86** (.86/.87)	**.76** (.75/.76)
🎯 M:AtMostOne2	**.86** (.86/.87)	**.76** (.75/.76)
🎯 M:Budget	.85 (.85/.86)	.75 (.75/.76)
Faithful		
🎲 Gumbel Matching	.85 (.84/.85)	.73 (.72/.73)
🎯 M:XorAtMostOne	.85 (.85/.85)	.73 (.72/.73)
🎯 M:AtMostOne2	.85 (.85/.85)	.73 (.73/.73)
🎯 M:Budget	.82 (.81/.82)	.68 (.67/.68)

Table 6: Model predictive performances across datasets, for stochastic (🎲) and deterministic (🎯) methods. We report mean and min/max F_1 scores across three random seeds on both SNLI and MNLI test sets. We bold the best-performing rationalized models for each corpus. † means results come from Swanson et al. (2020).

HANS Subcomponent	Vanilla	Augmented
Entailment		
Lexical Overlap	.9942	.9962
Subsequence	.9960	1.0
Constituent	.9988	1.0
Non-entailment		
Lexical Overlap	.0052	.9998
Subsequence	.0016	1.0
Constituent	.0122	1.0

Table 7: Model predictive performances for the vanilla and augmented models evaluated on the HANS evaluation set. We report accuracies for each of the six subcomponents of the evaluation set.

runs is also reflected on the token-level F_1 scores; and although the rationalizers that use sparse attention mechanisms are competitive with our proposed strategy, the lack of regularization on what comes to the rationale extraction leads to variable sized rationales which is also reflected on poorer matchings. We also observe that, when degeneration does not occur, HardKuma generally extracts high quality rationales on what comes to matching the human annotations. It is also worth remarking that the sparsemax and top-k strategies are not expected to fare well on this metric because human annotations for these datasets are at the *sentence-level*. Our strategy, however, not only pushes for sparser rationales but also encourages contiguity on the extraction.

5.3 Extraction of Text Matchings

Predictive Performance. We report the predictive performances of all models in Table 6. Both the strategies that use the LP-SparseMAP extraction layer and our proposed stochastic matchings extractor outperform the OT variants for matchings extraction. We observe that, contrary to the text highlights experiments, the stochastic matchings extraction model does not exhibit noticeably higher variability compared to the deterministic models. In general, the faithful models are competitive with the non-faithful models. Since the latter ones are constrained to only utilize information from the premise that comes from alignments, these results demonstrate the effectiveness of the alignment extraction. As expected, there is a slight trade-off between how constrained the alignment is and the model's predictive performance. This is more noticeable with the M:Budget strategy, the most constrained version of our proposed strategies, in the faithful scenario. We show examples of matchings extracted by SPECTRA in §H.

Heuristics Analysis with HANS. We used two different M:AtMostOne2 models for our analysis: a first one trained on MNLI (**Vanilla**), and a second one trained on MNLI augmented (**Augmented**) with 30,000 HANS-like examples ($\approx 8\%$ of MNLI original size), replicating the data augmentation scenario in (McCoy et al., 2019). We evaluated both models on the HANS evaluation set, which has six subcomponents, each defined by its correct label and the heuristic it addresses. We report the results in Table 7. Our models behave similarly to those in (McCoy et al., 2019): when we augment the training set with HANS-like examples, the model no longer associates the heuristics to entailment. By observation of the extracted matchings,

we noticed that these were similar between the two models. Thus, the effect of the augmented data resides on how the information from the matchings is used after the extraction layer. We show examples of matchings in §H.

6 Related Work

Selective Rationalization. There is a long string of work on interpreting predictions made by neural networks (Lipton, 2017; Doshi-Velez and Kim, 2017; Gilpin et al., 2019; Wiegreffe and Marasović, 2021; Zhang et al., 2021a). Our paper focus on selective rationalizers, which have been used for extraction of text highlights (Lei et al., 2016; Bastings et al., 2019; Yu et al., 2019; DeYoung et al., 2019; Treviso and Martins, 2020; Zhang et al., 2021b) and text matchings (Swanson et al., 2020). Most works rely on stochastic rationale generation or deterministic attention mechanisms, but the two approaches have never been extensively compared. Our work adds that comparison and contributes with an easy-to-train fully differentiable rationalizer that allows for flexible constrained rationale extraction. Our strategy for rationalization based on sparse structured prediction on factor graphs constitutes a unified framework for deterministic extraction of different structured rationales.

Structured Prediction on Factor Graphs. Kim et al. (2017) incorporate structured models in attention mechanisms as a way to model rich structural dependencies, leading to a dense probability distribution over structures. Niculae et al. (2018) propose SparseMAP, which yields a sparse probability distribution over structures and can be computed using calls to a MAP oracle, making it applicable to problems (e.g. matchings) for which marginal inference is intractable but MAP is not. However, the requirement of an exact MAP oracle prohibits its application for more expressive structured models such as loopy graphical models and logic constraints. This limitation is overcome by LP-SparseMAP (Niculae and Martins, 2020) via a local polytope relaxation, extending the previous method to sparse differentiable optimization in any factor graph with arbitrarily complex structure. While other relaxations for matchings – such as entropic regularization leading to Sinkhorn's algorithm (Cuturi, 2013b) – that are tractable and efficient exist and have been used for rationalization (Swanson et al., 2020), we use LP-SparseMAP for rationale extraction in our work. Our approach for rationalization focuses on learning and explaining with latent structure extracted by structured prediction on factor graphs.

Sentence Compression and Summarization. Work on sentence compression and summarization bears some resemblance to selective rationalization for text highlights extraction. Titov and McDonald (2008) propose a statistical model which is able to discover corresponding topics in text and extract informative snippets of text by predicting a stochastic mask via Gibbs sampling. McDonald (2006) proposes a budgeted dynamic program in the same vein as that of the H:SeqBudget strategy for text highlights extraction. Berg-Kirkpatrick et al. (2011) and Almeida and Martins (2013) propose models that jointly extract and compress sentences. Our work differs in that our setting is completely unsupervised and we need to differentiate through the extractive layers.

7 Conclusions

We have proposed SPECTRA, an easy-to-train fully differentiable rationalizer that allows for flexible constrained rationale extraction. We have provided a comparative study with stochastic and deterministic approaches for rationalization, showing that SPECTRA generally outperforms previous rationalizers in text classification and natural language inference tasks. Moreover, it does so while exhibiting less variability than stochastic methods and easing regularization of the rationale extraction when compared to previous deterministic approaches. Our framework constitutes a unified framework for deterministic extraction of different structured rationales. We hope that our work spurs future research on rationalization for different structured explanations.

Acknowledgements

We are grateful to Vlad Niculae for his valuable help and insight on LP-SparseMAP. We would like to thank Marcos Treviso for helping to start this project. We are grateful to Wilker Aziz, António Farinhas, Ben Peters, Gonçalo Correia, and the reviewers, for their helpful feedback and discussions. This work was supported by the European Research Council (ERC StG DeepSPIN 758969, by the FCT through contract UIDB/50008/2020, and by the P2020 programs MAIA and Unbabel4EU (LISBOA-01-0247-FEDER-045909 and LISBOA-01-0247-FEDER-042671).

References

Miguel Almeida and André Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 196–206, Sofia, Bulgaria. Association for Computational Linguistics.

Yujia Bao, Shiyu Chang, Mo Yu, and Regina Barzilay. 2018. Deriving machine attention from human rationales. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1903–1913, Brussels, Belgium. Association for Computational Linguistics.

Jasmijn Bastings, Wilker Aziz, and Ivan Titov. 2019. Interpretable neural predictions with differentiable binary variables. In *Proc. ACL*.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 481–490, Portland, Oregon, USA. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Shiyu Chang, Yang Zhang, Mo Yu, and Tommi S. Jaakkola. 2020. Invariant rationalization.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada. Association for Computational Linguistics.

Caio Corro and Ivan Titov. 2019a. Differentiable perturb-and-parse: Semi-supervised parsing with a structured variational autoencoder.

Caio Corro and Ivan Titov. 2019b. Learning latent trees with stochastic perturbations and differentiable dynamic programming. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5508–5521, Florence, Italy. Association for Computational Linguistics.

Marco Cuturi. 2013a. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Marco Cuturi. 2013b. Sinkhorn distances: Lightspeed computation of optimal transport. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 2292–2300, Red Hook, NY, USA. Curran Associates Inc.

Gianna Del Corso, Antonio Gulli, and Francesco Romani. 2005. Ranking a stream of news. pages 97–106.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C Wallace. 2019. Eraser: A benchmark to evaluate rationalized nlp models. *arXiv preprint arXiv:1911.03429*.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning.

Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2019. Explaining explanations: An overview of interpretability of machine learning.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Alon Jacovi and Yoav Goldberg. 2021. Aligning Faithful Interpretations with their Social Attribution. *Transactions of the Association for Computational Linguistics*, 9:294–310.

Sarthak Jain, Sarah Wiegreffe, Yuval Pinter, and Byron C Wallace. 2020. Learning to faithfully rationalize by construction. *arXiv preprint arXiv:2005.00115*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298, Cambridge, MA. Association for Computational Linguistics.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proc. EMNLP*, pages 107–117.

Zachary C. Lipton. 2017. The mythos of model interpretability.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

André F. T. Martins and Ramón Fernandez Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. *CoRR*, abs/1602.02068.

André FT Martins, Mário AT Figueiredo, Pedro MQ Aguiar, Noah A Smith, and Eric P Xing. 2015. Ad3: Alternating directions dual decomposition for map inference in graphical models. *The Journal of Machine Learning Research*, 16(1):495–545.

Julian McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning attitudes and attributes from multi-aspect reviews.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy. Association for Computational Linguistics.

Vlad Niculae and Mathieu Blondel. 2019. A regularized framework for sparse and structured neural attention.

Vlad Niculae and André F. T. Martins. 2020. Lp-sparsemap: Differentiable relaxed optimization for sparse structured prediction.

Vlad Niculae, André F. T. Martins, Mathieu Blondel, and Claire Cardie. 2018. Sparsemap: Differentiable sparse structured inference.

George Papandreou and Alan L. Yuille. 2011. Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models. In *2011 International Conference on Computer Vision*, pages 193–200.

Bhargavi Paranjape, Mandar Joshi, John Thickstun, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. An information bottleneck approach for controlling conciseness in rationale extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

L.R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Kyle Swanson, Lili Yu, and Tao Lei. 2020. Rationalizing text matching: Learning sparse alignments via optimal transport. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5609–5626, Online. Association for Computational Linguistics.

Ivan Titov and Ryan McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of ACL-08: HLT*, pages 308–316, Columbus, Ohio. Association for Computational Linguistics.

Marcos Treviso and André F. T. Martins. 2020. The explanation game: Towards prediction explainability through sparse communication. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 107–118, Online. Association for Computational Linguistics.

A. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

Martin J Wainwright and Michael Irwin Jordan. 2008. *Graphical models, exponential families, and variational inference*. Now Publishers Inc.

Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent aspect rating analysis on review text data: A rating regression approach. pages 783–792.

Sarah Wiegreffe and Ana Marasović. 2021. Teach me to explain: A review of datasets for explainable nlp.

R. J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Thomas Wolf, Quentin Lhoest, Patrick von Platen, Yacine Jernite, Mariama Drame, Julien Plu, Julien Chaumond, Clement Delangue, Clara Ma, Abhishek Thakur, Suraj Patil, Joe Davison, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angie McMillan-Major, Simon Brandeis, Sylvain Gugger,

François Lagunas, Lysandre Debut, Morgan Funtowicz, Anthony Moi, Sasha Rush, Philipp Schmidd, Pierric Cistac, Victor Muštar, Jeff Boudier, and Anna Tordjmann. 2020. Datasets. *GitHub. Note: https://github.com/huggingface/datasets*, 1.

Mo Yu, Shiyu Chang, Yang Zhang, and Tommi Jaakkola. 2019. Rethinking cooperative rationalization: Introspective extraction and complement control. In *Proc. EMNLP-IJCNLP*, pages 4085–4094.

Omar F. Zaidan, Jason Eisner, and Christine Piatko. 2007. Using "annotator rationales" to improve machine learning for text categorization. In *NAACL HLT 2007; Proceedings of the Main Conference*, pages 260–267.

Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. 2021a. A survey on neural network interpretability.

Zijian Zhang, Koustav Rudra, and Avishek Anand. 2021b. Explain and predict, and then predict again. *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*.

A Datasets for Highlights Extraction

We used five datasets for sentiment analysis: four for text classification (SST, AgNews, IMDB, Hotels) (Socher et al., 2013; Del Corso et al., 2005; Maas et al., 2011; Wang et al., 2010) and one for regression (BeerAdvocate) (McAuley et al., 2012). The Hotels and BeerAdvocate datasets contain data instances for multiple aspects. In this work, we use the Hotels' location aspect and the BeerAdvocate's appearance aspect. These two datasets contain *sentence-level* rationale annotations for their test sets. For these datasets, we use the splits used in Bao et al. (2018). For all other datasets, we use the splits in Wolf et al. (2020). For IMDB and AgNews we randomly selected 10%, 15% of examples from the training set to be used as validation data, respectively. Table 8 shows statistics for each dataset and rationale length as a percentage of each document.

Dataset	# Train	# Test	# Classes	Rationale Length (%)
SST	6920	1821	2	20
AgNews	120K	7600	4	20
IMDB	25K	25K	2	20
Hotels	12K	200	2	15
Beer	80K	997	–	10

Table 8: Dataset statistics and rationale length as a percentage of each document.

For the datasets without human annotations, we used the same sparsity level (20%) – Jain et al. (2020) uses this value for AgNews and SST; for BeerAdvocate, we used the sparsity levels used in Lei et al. (2016) and Yu et al. (2019); and, for Hotels we opted to select a sparsity level of 15% (human annotations average around 10% sparsity level).

B Datasets for Matchings Extraction

For natural language inference (NLI), we used SNLI and MNLI (Bowman et al., 2015; Chen et al., 2017). For MNLI, we split the MNLI matched validation set into equal validation and test sets. Table 9 shows statistics for each dataset and the alignment budget used for the M:Budget factor.

For SNLI, we set the Budget B to 4 to compare with the OT approach (OT exact $k = 4$) of Swanson et al. (2020). For MNLI, we set B to 6, since the average premise length in MNLI is around 50% bigger than that of SNLI.

Dataset	# Train	# Test	# Classes	Alignment Budget
SNLI	550K	10K	3	4
MNLI	392K	10K	3	6
HANS	30K	30K	2	–

Table 9: Dataset statistics and alignment budget for each dataset.

We also conduct experiments with the HANS (McCoy et al., 2019) dataset. This dataset consists of a controlled evaluation set to detect whether NLI systems are exploring linguistic heuristics such as lexical overlap, subsequence and constituent heuristics. A detailed description of each of these heuristics can be found in the original paper. The dataset is also constituted by 30,000 HANS-like examples that can be used to augment existing NLI training sets such as SNLI or MNLI.

C Implementation Details

C.1 Rationalizers Experimental Setup

For all rationalizers, we map each input word to 300D-pretrained GloVe embeddings from 840B release (Pennington et al., 2014) that are kept frozen. We instantiate all encoder networks as bidirectional LSTM (Hochreiter and Schmidhuber, 1997) layers (BiLSTM) (w/ hidden size 200) similarly to Lei et al. (2016); Bastings et al. (2019); Treviso and Martins (2020). Although other works (Jain et al., 2020; Paranjape et al., 2020) use more powerful BERT-based representations, we firstly experimented with BiLSTM layers and noticed our results were competitive with those reported in Jain et al. (2020). We used the Adam optimizer (Kingma and Ba, 2017) for all experiments with learning rate within $\{1 \times 10^{-3}, 5 \times 10^{-4}, 1 \times 10^{-4}, 5 \times 10^{-5}\}$ and ℓ_2 regularization within $\{10^{-4}, 10^{-5}\}$. We also enforce a grad norm of 5.0. We train all models for highlights extraction for a minimum of 5 epochs and maximum of 25 epochs. For matchings extraction, we set the number of minimum epochs and maximum epochs to 3 and 10, respectively.

Training for all methods for highlights extraction but HardKuma is stopped if Macro F_1 (for classification) or MSE (for regression) is not improved for 5 epochs. For matchings extraction, training is stopped if Macro F_1 does not improve for 3 epochs. For HardKuma, we train until the maximum number of epochs. This is because the rationale length might vary considerably during training due to the Lagrangian relaxation algorithm that is employed at training time. We found that using early stopping would often favour models that selected almost all of the input text. Unlike Jain et al. (2020), we decided to carefully tune both model and the Lagrangian relaxation algorithm hyperparameters for this rationalizer. This had a big impact on the performance, as HardKuma performed poorly with the top-k and contiguous strategies at inference time. Even though some careful tuning is required and degeneration might occur for some random seeds, it is still much less cumbersome than tuning the variants of the rationalizer of Lei et al. (2016). We hypothesize that this is mostly due to two factors: the control on the rationale average size that the Lagrangian relaxation algorithm aims to impose; and the gradient estimates with reparameterized gradients exhibit less variance than those with the score function estimator.

All models for highlights extraction have 1.8M trainable parameters. Models for faithful and non-faithful selective rationalization of text matchings have 1.7M and 1.8M trainable parameters, respectively.

C.2 SPECTRA Sparsity Regularization

During training, we apply a temperature term T in the sparsemax and fusedmax operators. This parameter is set within $\{0.05, 0.1, 0.2\}$. The total variation regularization for fusedmax is set to 0.7.

For the models that use the LP-SparseMAP extraction layer, we use a temperature term T set within $\{0.05, 0.1, 0.2\}$ during training. Moreover, for the H:SeqBudget, we set the transition scores within $\{0.001, 0.005\}$ for all datasets. All hyperparamter searches were conducted manually.

The LP-SparseMAP problem can be interpreted as the ℓ_2-regularized LP-MAP. Its output corresponds to a probability distribution over a sparse set of structures. Therefore LP-MAP can be seen as LP-SparseMAP with the scores divided by a zero-limit temperature parameter. This procedure at test time would lead to the LP-MAP solution, which is generally an outer relaxation of MAP (Martins et al., 2015). When inference in the factor graph is exact, the solutions of the LP-MAP are integer (i.e., LP-MAP yields the true MAP). But that is not the case for when inference in the factor graph is not exact. Thus, LP-SparseMAP solutions for this test time setting might be a soft or discrete selection of parts of the input. We used a temperature parameter

of 10^{-3} at validation and testing time.

C.3 Computing Infrastructure

Our infrastructure consists of 2 machines with the specifications shown in Table 10. The machines were used interchangeably, and all experiments were executed in a single GPU. We did not observe significant differences in the execution time of our models across different machines.

#	GPU	CPU
1	4×GTX 1080 Ti - 12GB	8×Intel i7-9800X @ 3.80GHz - 128GB
2	4×RTX 2080 Ti - 12GB	48xIntel Xeon Silver @ 2.20GHz - 128GB

Table 10: Computing infrastructure used to run our experiments.

D Full-text Baselines

Table 11 shows the performance scores of each classifier on the test sets for highlights extraction when feeding the full input document to the predictor.

Dataset	Performance Score
SST	0.83 (.82/.83)
AgNews	0.93 (.93/.93)
IMDB	0.90 (.90/.90)
Beer	0.019 (.18/.21)
Hotels	0.87 (.86/.88)

Table 11: Model predictive performances across datasets using full-text. We report mean and min/max F_1 scores across five random seeds on test sets for all datasets but Beer where we report MSE.

Table 12 shows the performance scores of the ESIM model for both SNLI and MNLI.

Dataset	Performance Score
SNLI	0.86 (.86/.86)
MNLI	0.74 (.73/.74)

Table 12: ESIM predictive performances for SNLI and MNLI. We report mean and min/max F_1 scores across three random seeds on test sets for both datasets.

E Computational Cost of SPECTRA

Tables 13 and 14 show the average training and validation time per epoch for each dataset with the SPECTRA strategies for highlights and matchings extraction, respectively. We also present, for comparison, the average training and validation time per epoch for some of the other methods we used in the paper. The batch size for the models for highlights extraction is 32. For matchings extraction, the batch size for all models but M:Budget is 16. For M:Budget, the batch size is 8.

The computational time of SPECTRA depends on several factors inherited from the use of LP-SparseMAP as the extractive method. Generally, the bigger the number of local factors $f \in \mathcal{F}$, the more costly it is to compute a solution. Thus, it might be necessary to increase the number of iterations for the LP-SparseMAP to converge to a solution for which all factors agree. We set this number to 10 in training time following Niculae and Martins (2020). During inference, we set a maximum number of iterations of 1000. For highlights extraction, the H:SeqBudget consists of a single factor, thus the solution is found within a single iteration. For matchings extraction, our factors consist of multiple local factors that impose hard constraints that must agree in the final matching: M:XorAtMostOne and M:AtMostOne2 consist of $L_P + L_H$ local factors, and M:Budget adds an additional global budget factor to the factor graph of M:AtMostOne2, yielding a more complex overall problem. Faster times would be achieved for smaller values of maximum number of iterations.

Strategy	SST	AgNews	IMDB	Beer	Hotels
Training Time (in seconds)					
SPECTRA	20	300	600	85	550
HardKuma	18	180	120	75	220
SFE	10	165	120	40	200
Sparsemax	15	165	120	40	200
Validation Time (in seconds)					
SPECTRA	2	50	45	10	60
HardKuma	1	25	15	6	30
SFE	1	25	15	6	30
Sparsemax	1	20	15	6	30

Table 13: Average training and validation time per epoch respective to some strategies used in the paper for each dataset used for highlights extraction.

F SPECTRA performance for varying rationale length/alignment budget

To analyse how SPECTRA fares for different sparsity levels, we ran experiments for highlights extraction and matchings extraction (see Table 15 and Table 16) for different budget values.

Strategy	SNLI	MNLI
Training Time (in minutes)		
ESIM	26	23
M:XorAtMostOne	53	60
M:AtMostOne2	53	60
M:Budget	56	65
Gumbel	28	23
Validation Time (in seconds)		
ESIM	10	10
M:XorAtMostOne	65	60
M:AtMostOne2	65	70
M:Budget	110	115
Gumbel	10	10

Table 14: Average training and validation time per epoch for each dataset and each strategy used for matchings extraction.

B	SST ↑	AgNews ↑	IMDB ↑	Beer ↓	Hotels ↑
2	0.578	0.870	0.891	0.020	0.875
5	0.726	0.903	0.900	0.019	0.867
10	0.799	0.915	0.891	0.016	0.890
15	0.798	0.916	0.893	0.016	0.906
20	0.803	0.918	0.891	0.017	0.895

Table 15: Model predictive performances across datasets and different budget values B for the SPECTRA method for highlights extraction. We report F_1 scores on test sets for all datasets but Beer where we report MSE. These results are respective to one random seed.

Budget B	SNLI ↑	MNLI ↑
4	0.857	0.731
5	0.851	0.731
6	0.852	0.755

Table 16: Model predictive performances across datasets and different budget values for the SPECTRA method for matchings extraction. We report F_1 scores on test sets for all datasets. These results are respective to one random seed.

G Highlights Extracted with SPECTRA

Figure 1 shows examples of highlights extracted by SPECTRA model on the AgNews and Beer dataset. Interestingly, when compared to human annotations on the Beer dataset, we notice that SPECTRA usually disregards highlighting stopwords. While these explanations do not lose relevant meaning when compared to the human explanations, this ultimately slightly hinders the performance on the matching with human annotations.

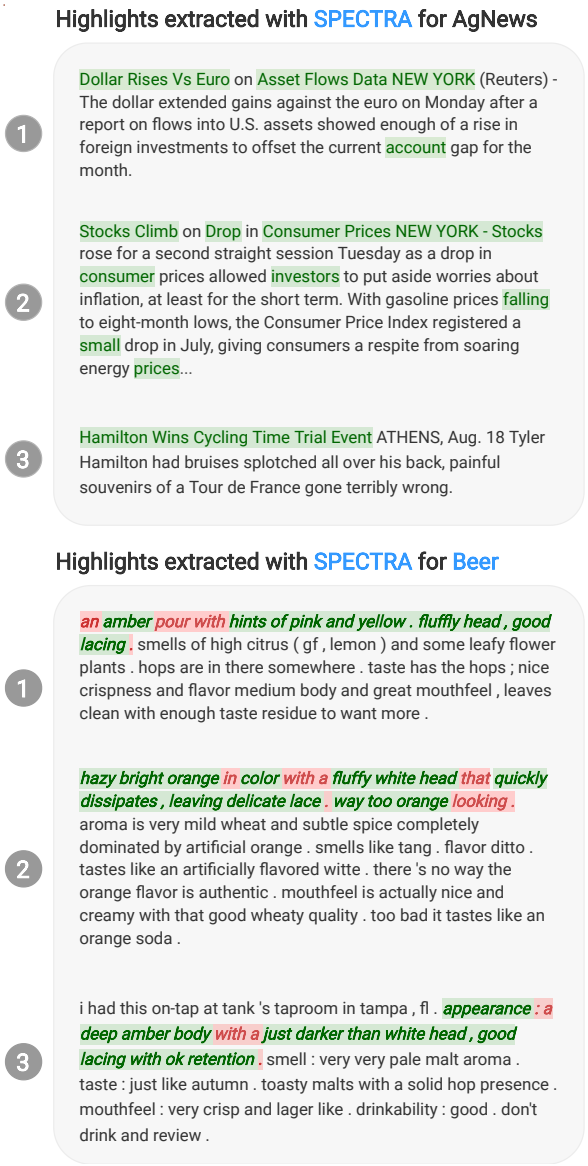

Figure 1: Examples of extracted highlights (green shaded input tokens) with SPECTRA for AgNews and Beer documents. For the rationales with Beer, we show the human annotations in bold and italic (we shade in red the mismatches with the human annotations).

H Matchings Extracted with SPECTRA

Synthetic Matchings. In Figure 2 we show the extracted matchings with the three different SPECTRA factors that we used in the paper for a synthetic score matrix. The M:XOR-AtMostOne factor constraints the alignment matrix $\boldsymbol{Z} \in \mathbb{R}^{L_P \times L_H}$ to be such that for each line i in $\boldsymbol{Z}$, we have $\sum_{n=1}^{L_H} z_{in} = 1$. For M:AtMostOne2 we have that for each line i in $\boldsymbol{Z}$, $\sum_{n=1}^{L_H} z_{in} \leq 1$. And, finally, the more constrained version of M:Budget is such that for each line i in $\boldsymbol{Z}$, we have $\sum_{n=1}^{L_H} z_{in} \leq B$, in which B is the Budget value.

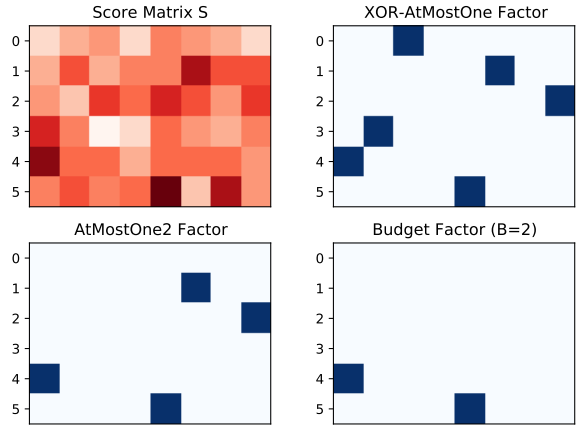

Figure 2: Extracted matchings with the SPECTRA strategies for a synthetic 6×8 matrix.

Examples extracted from SNLI. We show in Figures 3, 4 and 5 some examples of matchings extracted with the three different SPECTRA strategies on the SNLI dataset. For these examples all non-null entries in $\boldsymbol{Z}$ have value 1.

Examples extracted from HANS. We show in Figure 6 examples of matchings extracted with SPECTRA for the model trained on MNLI augmented with HANS-like examples (Augmented). For all these examples, the original MNLI model without augmentation (Vanilla) classified the examples as entailment, whereas the Augmented model correctly classified them as non-entailment. Interestingly, the obtained matchings highlight the use of the heuristics that HANS aims to target. However, the Augmented model is able to process the information from the matchings in such a way that it correctly classifies most non-entailment examples (see Table 7).

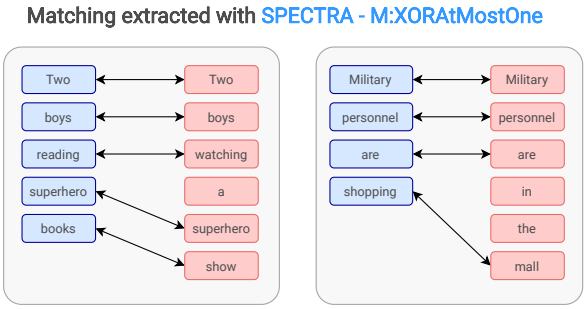

Figure 3: Examples of extracted matchings with M:XORAtMostOne. The premise is shown on the left and the hypothesis is shown on the right.

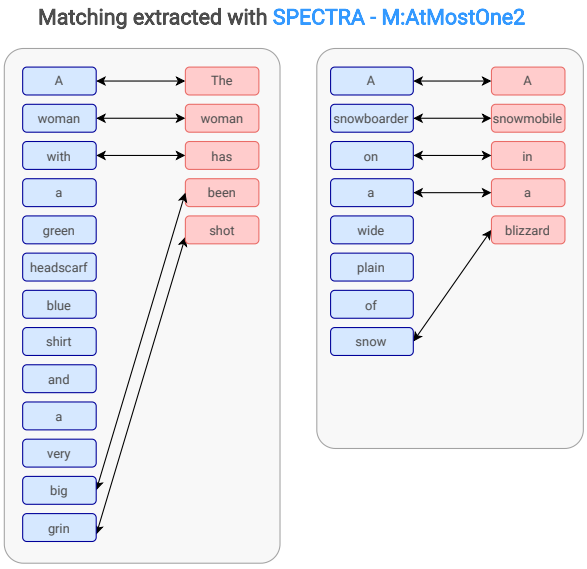

Figure 4: Examples of extracted matchings with M:AtMostOne2. The premise is shown on the left and the hypothesis is shown on the right.

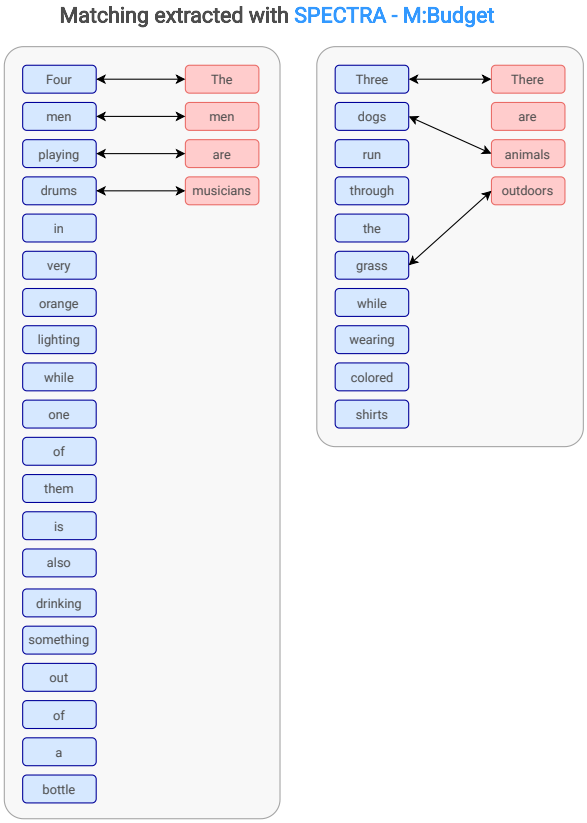

Figure 5: Examples of extracted matchings with M:Budget – the Budget value is set to 4. The premise is shown on the left and the hypothesis is shown on the right.

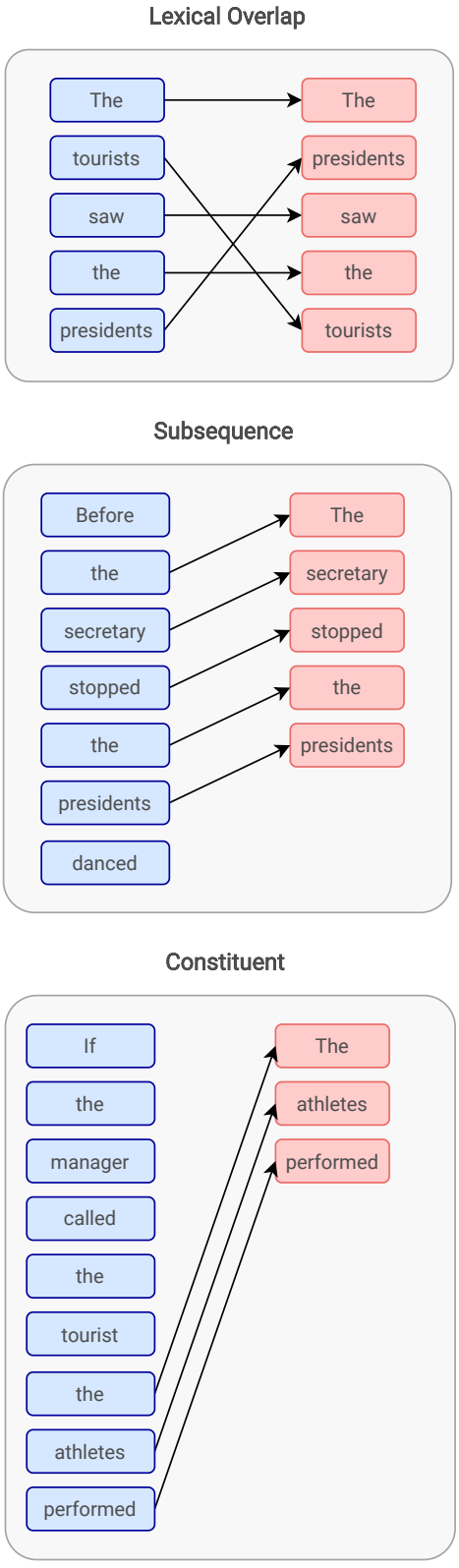

Figure 6: Examples of extracted matchings with SPECTRA (Augmented) that highlight the three linguistic heuristics of HANS: lexical overlap, constituent and subsequence heuristics. The premise is shown on the left and the hypothesis is shown on the right.