Canonical Correlation Inference for Mapping Abstract Scenes to Text

Citation for published version:

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Canonical Correlation Inference for Mapping Abstract Scenes to Text

Nikos Papasartantopoulos
School of Informatics
University of Edinburgh
nikos.papasa@ed.ac.uk

Helen Jiang
Department of Computer Science
Stanford University
helennn@stanford.edu

Shay B. Cohen
School of Informatics
University of Edinburgh
scohen@inf.ed.ac.uk

1 Introduction

Canonical correlation analysis (CCA) is a method to reduce the dimensionality of multiview data, introduced by Hotelling (1935). It takes two random vectors and computes their corresponding empirical cross-covariance matrix. It then applies singular value decomposition (SVD) on this matrix to get linear projections of the random vectors that have maximal correlation.

In this paper, we investigate the idea of using CCA for a full-fledged structured prediction problem. More specifically, we suggest a method in which we take a structured prediction problem training set, and then project both the inputs and the outputs to low-dimensional space. The projection ensures that inputs and outputs that correspond to each other are projected to close points in low-dimensional space. Decoding happens in the low-dimensional space. As such, our training algorithm builds on previous work by Udupa and Khapra (2010) and Jagarlamudi and Daumé III (2012) who used CCA for transliteration.

Our approach of canonical correlation inference is simple to implement and does not require complex engineering tailored to the task. It mainly needs two feature functions, one for the input values and one for the output values and does not require features combining the two. We also propose a simple decoding algorithm when the output space is text.

We test our learning algorithm on the domain of language and vision. We use the abstract scene dataset of Zitnick and Parikh (2013), with the goal of mapping images (in the form of clipart abstract scenes) to their corresponding image descriptions. This problem has a strong relationship to recent work in language and vision that has used neural networks or other computer vision techniques to solve a similar problem for real images (Section 2.2). Our work is most closely related to the work by Ortiz, Wolff, and Lapata (2015) who used phrase-based machine translation to translate images to corresponding descriptions.

2 Background and Notation

We give in this section some background information about CCA and the problem which we aim to solve with it.

2.1 Canonical Correlation Analysis

Canonical correlation analysis (CCA; Hotelling, 1935) is a technique for multiview dimensionality reduction, related to co-training (Blum and Mitchell 1998). CCA assumes that there are two views for a given set of data, where these two views are represented by two random vectors \( X \in \mathbb{R}^d \) and \( Y \in \mathbb{R}^d \).

The procedure that CCA follows finds a projection of the two views in a shared space of dimension \( m \), such that the correlation between the two views is maximized at each coordinate, and there is minimal redundancy between the coordinates of each view. This means that CCA solves the following sequence of optimization problems for \( j \in \{1, \ldots, m\} \) where \( a_j \in \mathbb{R}^{1 \times d} \) and \( b_j \in \mathbb{R}^{1 \times d} \):

\[
\arg\max_{a_j,b_j} \text{corr}(a_j X^T, b_j Y^T)
\]

such that \( \text{corr}(a_j X^T, a_k X^T) = 0, \quad k < j \)

\( \text{corr}(b_j Y^T, b_k Y^T) = 0, \quad k < j \)

where \( \text{corr} \) is a function that accepts two vectors and returns the Pearson correlation between the pairwise elements of the two vectors. The problem of CCA can be solved by applying singular value decomposition (SVD) on a cross-covariance matrix between the two random vectors \( X \) and \( Y \), normalized by the covariance matrices of \( X \) and \( Y \).

More specifically, CCA is solved by applying thin singular value decomposition (SVD) on the empirical version of the following matrix:

\[
(E[XX^T])^{-\frac{1}{2}} E[XY^T] (E[YY^T])^{-\frac{1}{2}} \approx U \Sigma V^T,
\]

where \( E[\cdot] \) is the expectation operator and \( \Sigma \) is a diagonal matrix of size \( m \times m \) for some small \( m \). Since this version
of CCA requires inverting matrices of potentially large dimension \((d \times d)\) and \((d' \times d')\), it is often the case that only the diagonal elements from these matrices are used, as we see in Section 3.

CCA and its variants have been used in various contexts in NLP. They were used to derive word embeddings (Dhillon, Foster, and Ungar 2015), derive multilingual embeddings (Faruqui and Dyer 2014; Lu et al. 2015), build bilingual lexicons (Haghighi et al. 2008), encode prior knowledge into embeddings (Osborne, Narayan, and Cohen 2016), semantically analyze text (Vinokourov, Cristiani, and Shawe-Taylor 2002) and reduce the dimensions of text with many views (Rastogi, Van Durme, and Arora 2015). CCA is also an important sub-routine in the family of spectral algorithms for estimating structured models such as latent-variable PCFGs and HMMs (Cohen et al. 2012; Stratos, Collins, and Hsu 2016) or finding word clusters (Stratos et al. 2014). Variants of it have been developed, such as DeepCCA (Andrew et al. 2013).

2.2 Describing Images

Image description, the task of assigning textual descriptions to images, is a problem that has been thoroughly studied in various setups and variances. Usually, proposed methods treat images as sets of objects identified in them (bags of regions), however there has been work that uses some kind of structural image cues or relations. An excellent example of such cues are visual dependency representations (Elliott and Keller 2013), which can be used to outline what can be described as the visual counterpart of dependency trees.

Common is also the idea of solving a related but slightly different problem, the one of matching sentences to images using existing descriptions. The core of those approaches is an information retrieval task, where for every novel image, a set of similar images is retrieved and generation proceeds using the descriptions of those images. Search queries are posed against a visual space (Ordonez, Kulkarni, and Berg 2011; Mason and Charniak 2014) or a multimodal space, where images and descriptions have been projected (Farhadi et al. 2010; Hodosh, Young, and Hockenmaier 2013). Instead of whole sentences, phrases from existing human generated descriptions have also been used (Kuznetsova et al. 2012).

Approaches to image description generation have for a long time relied on a set of predefined sentence templates (Kulkarni et al. 2011; Elliott and Keller 2013; Yang et al. 2011) or used syntactic trees (Mitchell et al. 2012), while more recently, methods that use neural models (Kiros, Salakhutdinov, and Zemel 2014; Vinyals et al. 2015) have appeared, that avoid the use of any kind of predefined pattern. Approaches like the latter follow the paradigm of tackling the problem as an end-to-end optimization problem. Ortiz, Wolff, and Lapata (2015) describe a two-step process: a content selection phase, where the objects that need to be described are picked, and then the text realization, where the description is generated by employing a statistical machine translation (MT) system.

While computer vision advances have given an unprecedented potential to image description generation, vision performance affects the generation process, as those two problems are commonly solved together in a pipeline or a joint fashion. To countermeasure that, Zitnick and Parikh (2013) introduced the notion of “abstract scenes”, that is abstract images generated by stitching together clipart images. Their intuition is that working on abstract scenes can allow for a more clean and isolated evaluation of caption generators and also lead to relatively easy construction of datasets of images with semantically similar content. An example of such dataset is the Abstract Scenes Dataset.\(^1\) This dataset has been used for description generation (Ortiz, Wolff, and Lapata 2015), sentence-to-scene generation (Zitnick, Parikh, and Vanderwende 2013) and object dynamics prediction (Fouhey and Zitnick 2014) so far.

3 Learning and Decoding

We now describe the learning algorithm, based on CCA, and the corresponding decoding algorithm when the output space is text.

3.1 Learning Based on Canonical Correlation Analysis

We assume two structured spaces, an input space \(X\) and an output space \(Y\). As usual in the supervised setting, we are given a set of instances \((x_1, y_1), \ldots, (x_n, y_n) \in X \times Y\), and the goal is to learn a decoder \(\text{dec}: X \rightarrow Y\) such that \(\text{dec}(x)\) is the “correct” output as learned based on the training examples. The basic idea in our learning procedure is to learn two projection functions \(u: X \rightarrow \mathbb{R}^m\) and \(v: Y \rightarrow \mathbb{R}^m\) for some low-dimensional \(m\) (relatively to \(d\) and \(d'\)). In addition, we assume the existence of a similarity measure \(\rho: \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}\) such that for any \(x, y\), the better \(y\) “matches” the \(x\) according to the training data, the larger \(\rho(u(x), v(y))\) is.

The decoder \(\text{dec}(x)\) is then defined as:

\[
\text{dec}(x) = \arg \max_{y \in Y} \rho(u(x), v(y)).
\]

Our key observation is that one can use canonical correlation analysis to learn the two projections \(u\) and \(v\). This is similar to the observation made by Udupa and Khapra (2010) and Jagarlamudi and Daumé III (2012) in previous work about transliteration. The learning algorithm assumes the existence of two feature functions \(\phi: X \rightarrow \mathbb{R}^{d \times 1}\) and \(\psi: Y \rightarrow \mathbb{R}^{d' \times 1}\), where \(d\) and \(d'\) could potentially be large, and the feature functions could potentially lead to sparse vectors.

We then apply a modified version of canonical correlation analysis on the two “views”: one view corresponds to the input feature function and the other view corresponds to the output feature function. This means we calculate the following three matrices \(D_1 \in \mathbb{R}^{d \times d}, D_2 \in \mathbb{R}^{d' \times d'}\) and \(\Omega \in \mathbb{R}^{d \times d'}:\n
\(^1\)https://vision.ece.vt.edu/clipart/
Inputs: Set of examples \((x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\) for \(i \in \{1, \ldots, n\}\). An integer \(m\). Two feature functions \(\phi(x)\) and \(\psi(y)\).

Data structures:
Projection matrices \(U\) and \(V\).

Algorithm:
(Cross-covariance estimation)
- Calculate \(\Omega \in \mathbb{R}^{d \times d'}\)
  \[\Omega_{ij} = \sum_{k=1}^{n} \phi(x_k)_i [\psi(y_k)]_j\]
- Calculate \(D_1 \in \mathbb{R}^{d \times d}\) such that \((D_1)_{ij} = 0\) for \(i \neq j\) and
  \[(D_1)_{ii} = \sum_{k=1}^{n} [\phi(x_k)_i [\phi(x_k)]_i]\]
- Calculate \(D_2 \in \mathbb{R}^{d' \times d'}\) such that \((D_2)_{ij} = 0\) for \(i \neq j\) and
  \[(D_2)_{ii} = \sum_{k=1}^{n} [\psi(y_k)_i [\psi(y_k)]_i]\]

(Singular value decomposition step)
Calculate \(m\)-rank thin SVD on \(D_1^{-\frac{1}{2}} \Omega D_2^{-\frac{1}{2}}\). Let \(U\) and \(V\) be the two resulting projection matrices. Return the two functions \(u(x) = (D_1^{-\frac{1}{2}} U)^\top \phi(x)\) and \(v(y) = (D_2^{-\frac{1}{2}} V)^\top \psi(y)\).

Figure 1: The CCA learning algorithm.

For the similarity metric, we use the cosine similarity:
\[
\rho(z, z') = \frac{\sum_{i=1}^{m} z_i z'_i}{\sqrt{\sum_{i=1}^{m} z_i^2} \sqrt{\sum_{i=1}^{m} (z'_i)^2}} = \frac{\langle z, z' \rangle}{||z|| \cdot ||z'||}. 
\]

Figure 2 describes a sketch of our CCA inference algorithm.

Motivation
What is the motivation behind this use of CCA and the chosen projection matrices and similarity metric? Osborne, Narayan, and Cohen (2016) showed that CCA maximizes the following objective:
\[
\sum_{i,j} d_{ij} - n \sum_{i=1}^{n} d_{ii}^2,
\]
where
\[
d_{ij} = \sqrt{\frac{1}{2} \left( \sum_{k=1}^{m} (u(x_i) - v(y_j))^2 \right)}.
\]

The objective is maximized with respect to the projections that CCA finds, \(u\) and \(v\). This means that CCA finds projections such that the Euclidean distance between \(u(x)\) and \(v(y)\) for matching \(x\) and \(y\) is minimized, while it is maximized for \(x\) and \(y\) that have a mismatch between them.

As such, it is well-motivated to use a similarity metric \(\rho(u(x), v(y))\) which is inversely monotone with respect to the Euclidean distance between \(u(x)\) and \(v(y)\). We next note that for any two vectors \(z\) (denoting \(u(x)\)) and \(z'\) (denoting \(v(y)\)) it holds that (by simple algebraic manipulation):
\[
-\langle z, z' \rangle = \frac{1}{2} \left( ||z - z'||^2 - ||z||^2 - ||z'||^2 \right). \tag{2}
\]

This means that if the norms of \(z\) and \(z'\) are constant, then maximizing the cosine similarity between \(z\) and \(z'\) is the same as minimizing Euclidean distance between \(z\) and \(z'\). In our case, the norms of \(u(x)\) and \(v(y)\) are not constant, but we find that our algorithm is much more stable when the cosine similarity is used instead of the Euclidean distance.

We also note that in order to minimize the distance between \(z\) and \(z'\) to follow CCA, according to Eq. 2, we need to maximize the dot product between \(z\) and \(z'\) while minimizing the norm of \(z\) and \(z'\). This is indeed the recipe that the cosine similarity metric follows.

In Section 3.2 we give an additional interpretation to the use of cosine similarity, as finding the maximum aposteriori solution for a re-normalized von Mises-Fisher distribution.

3.2 When the Output Space is Language

While our approach to mapping from an input space to an output space through CCA is rather abstract and general, in this paper we focus in cases where the output space \(\mathcal{Y} \subseteq \Lambda^*\) is a set of strings over some alphabet \(\Lambda\), for example, the
The form sine similarity between $u$ (via the accept-rejection step is the exponentiated value of the co-

a given $x$ than that of any

new phrase such that in the training data, there is an occurrence of the

$P(v)$ relative frequency count to estimate

Wolff, and Lapata (2015), extracted using Moses, and use

In our experiments, we use the phrase table used by Ortiz,

in between these two words ($\ell$)

of times each phrase

appear between the context words ($y_k$). It also also holds that $\rho(u(x), v(y)) = \cos \theta$.

English language. For example, $Y$ could be the set of all $n$-

gram chains possible over some $n$-gram set or the set of possible composition of atomic phrases, similar to phrase tables

in phrase-based machine translation (Koehn et al. 2007).

For the language-vision problem we address in Section 4,

we assume the existence of a phrase table $P$, such that every

$y \in Y$ can be decomposed into a sequence of consecutive phrases $p_1, \ldots, p_r \in P$.

The problem of decoding over this space is not trivial.

This is regardless of $X$ – once $x$ is given, it is mapped using

$u(x)$ to a vector in $\mathbb{R}^m$, and at this point this is the information we use to further decode into $y$ – the structure of $X$

before this transformation does not change much the complexity of the problem. We propose the following approximate decoding algorithm $\text{dec}(x)$ for Eq. 1. The algorithm

is a Metropolis-Hastings (MH) algorithm that assumes the existence of a blackbox sampler $q(y \mid y')$ – the proposal distribution. This blackbox sampler randomly chooses two endpoints $k$ and $\ell$ in $y'$ and if possible, replaces all the words in these two words ($y_k' \cdots y_{\ell}'$) with a phrase $p \in P$

such that in the training data, there is an occurrence of the

new phrase $p$ after the word $y_{k-1}'$ and before the word $y_{\ell+1}'$.

As such, we are required to create a probabilistic table of the form $Q$:

$\Lambda \times \Lambda 
\rightarrow \mathbb{R}$

that maps a pair of words $y, y' \in \Lambda$ and a phrase $p \in P$ to the probability $Q(p \mid y, y')$.

In our experiments, we use the phrase table used by Ortiz,

Wolff, and Lapata (2015), extracted using Moses, and use relative frequency count to estimate $Q$: we count the number of times each phrase $p$ appears between the context words $y$ and $y'$ and normalize.

Since we are interested in maximizing the cosine similarity

between $v(y)$ and $u(x)$, after each sampling step, we

check whether the cosine similarity of the new $y$ is higher

(regardless of whether it is being accepted or rejected by the

MH algorithm) than that of any $y$ so far. We return the best

$y$ sampled.

The “true” unnormalized distribution we use in the

accept-rejection step is the exponentiated value of the cosine similarity between $u(x)$ and $v(y)$. This means that for a given $x$, the MH algorithm implicitly samples from the following distribution $P$:

$$P(y \mid x) = \frac{\exp \left( \frac{\langle u(x), v(y) \rangle}{\|u(x)\| \cdot \|v(y)\|} \right)}{Z(x)}$$

(3)

where

\begin{align*}
\alpha_0 &= \frac{\exp \left( \frac{1}{T} \rho(u(x), v(y)) + \eta|y| \right)}{\exp \left( \frac{1}{T} \rho(u(x), v(y')) + \eta|y'| \right)} \\
\alpha_1 &= \frac{|y|^2 Q(y_1 \cdots y_j \mid y_{l-1}, y_{j+1})}{|y'|^2 Q(y_1 \cdots y_j \mid y_{l-1}, y_{j+1})} \\
\alpha &= \{1, \alpha_0 \cdot \alpha_1\}.
\end{align*}

Uniformly sample a number from $[0, 1]$, and if it is smaller than $\alpha$, set $y'$ to be $y$.

Let $t \leftarrow \tau t$.

Return $y'$.
\[ Z(x) = \sum_{y' \in Y} \exp \left( \frac{\langle u(x), v(y') \rangle}{||u(x)|| \cdot ||v(y')||} \right). \]

The probability distribution \( P \) has a strong relationship to the von Mises-Fisher distribution, which is defined over vectors of unit vector. The von Mises-Fisher distribution has a parametric density function \( f(z; \mu) \) which is proportional to the exponentiated dot product between the unit vector \( z \) and some other unit vector \( \mu \) which serves as the parameter for the distribution. The main difference between the von Mises-Fisher distribution and the distribution defined in Eq. 3 is that we do not allow any unit vector to be used as \( v(y) \) – only those which originate in some output structure \( y \). As such, the distribution in Eq. 3 is a re-normalized version of the von-Mises distribution, after elements from its support are removed.

In a set of preliminary experiments, we found that while our algorithm gives adequate descriptions to the images, it is not unusual for it to give short descriptions that just mention a single object in the image. This relates to the adequacy-frequency tension that exists in machine translation problems. To overcome this issue, we add to the cosine similarity a term \( \eta ||y|| \) where \( \eta \) is some positive constant tuned on a development set. This pushes the decoding algorithm to prefer textual descriptions which are longer.

**Simulated Annealing** Since we are not interested in sampling from the distribution \( P(y \mid x) \), but actually find its mode, we use simulated annealing with our MH sampler.

This means that we exponentiate by a \( \frac{1}{t} \) term the unnormalized distribution we sample from, and decrease this temperature \( t \) as the sampler advances. We start with a temperature \( T = 10,000 \), and multiply \( t \) by \( \tau = 0.995 \) at each step. The idea is for the sampler to start with an exploratory phase, where it is jumping from different parts of the search space to others. As the temperature decreases, the sampler makes smaller jumps with the hope that it has gotten closer to parts of the search space where most of the probability mass is concentrated.

4 Experiments

Our experiments are performed on a language-vision dataset, with the goal of taking a so-called “abstract scene” (Zitnick and Parikh 2013) and finding a suitable textual description. Figure 2 gives a description of our CCA algorithm in the context of this problem.

The Abstract Scenes Dataset consists of 10,020 scenes, each represented as a set of clipart objects placed in different positions and sizes in a background image (consisting of a grassy area and sky). Cliparts can appear in different ways, for example, the boy and the girl (cliparts 18 and 19), can be depicted sad, angry, sitting or running. The descriptions were given using crowdsourcing.

We use the same data split as Ortiz, Wolff, and Lapata (2015). We use 7,014 of the scenes as a training set, 1,002 as a development set and 2,004 as a test set. Each scene is labeled with at most eight short captions. We use all of these captions in the training set, leading to a total of 42,276 training instances. We also use these captions as reference captions for both the development and the test set.

The feature function \( \phi(x) \) for an image is based on the “visual features” that come with the abstract scene dataset. More specifically, there are binary features that fire for 11 object categories, 58 specific objects, co-occurrence of object category pairs, co-occurrence of object instance pairs, absolute location of object categories and instances, absolute depth, relative location of objects, relative location with directionality the object is facing, a feature indicating whether an object is near a child’s hand or a child’s head and attributes of the children (pose and facial expression). The total number of features for this \( \phi \) function is 7,149. See more in the description of the abstract scene dataset.

The feature function \( \psi(y) \) for an image description is defined as a one-hot representation for all phrases from the phrase table of Ortiz, Wolff, and Lapata (2015) that fire in the image (the phrase table is denoted by \( P \) in Section 3.2). This phrase table was obtained through the Moses toolkit (Koehn et al. 2007). The total number of phrases in this phrase table is 30,911. The size of the domain of \( Q \) (meaning, the size of the phrase table with context words) is 120,019. Table 1 gives a few example phrases and their corresponding probabilities.

In our CCA learning algorithm, we also need to decide on the value of \( m \). We varied \( m \) between 30 and 300 (in steps of 10) and tuned its value on the development set by maximizing BLEU score against the set of references. Interestingly enough, the BLEU scores did not change that much (they usually were within one point of each other for sufficiently large \( m \)), pointing to a stability of the algorithm with respect to the number of dimensions used.

Ortiz, Wolff, and Lapata (2015) partially measure the success of their system by comparing BLEU and METEOR scores of their different systems while using the descriptions given in the dataset as a reference set. The scores for their

<table>
<thead>
<tr>
<th>( y )</th>
<th>( p )</th>
<th>( y' )</th>
<th>prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>waiting</td>
<td>to</td>
<td>get</td>
<td>1.000</td>
</tr>
<tr>
<td>with</td>
<td>the</td>
<td>bucket.</td>
<td>0.750</td>
</tr>
<tr>
<td>pizza</td>
<td>on</td>
<td>the</td>
<td>0.343</td>
</tr>
<tr>
<td>trying</td>
<td>to get away</td>
<td>from</td>
<td>jenny</td>
</tr>
<tr>
<td>baseball</td>
<td>with</td>
<td>the</td>
<td>0.011</td>
</tr>
<tr>
<td>is</td>
<td>playing near the</td>
<td>swings.</td>
<td>0.008</td>
</tr>
<tr>
<td>(begin) jenny is playing with a</td>
<td>colorful</td>
<td></td>
<td>0.033</td>
</tr>
<tr>
<td>is</td>
<td>surprised by the</td>
<td>owl</td>
<td>0.006</td>
</tr>
<tr>
<td>mike</td>
<td>and</td>
<td>the bear are</td>
<td>standing</td>
</tr>
</tbody>
</table>

Table 1: Example of phrases and their probabilities learned for the function \( Q(p \mid y,y') \). The marker (begin) marks the beginning of a sentence.

---

2Our dataset splits and other information can be found in [http://cohort.inf.ed.ac.uk/canonical-correlation-inference.html](http://cohort.inf.ed.ac.uk/canonical-correlation-inference.html).

3We use the multeval package from [https://github.com/jhclark/multeval](https://github.com/jhclark/multeval).
different systems are given in Table 2. They compare their system (SMT, based on phrase-based machine translation) against several baselines:

- **LBL**: a log-bilinear language model trained on the image captions only.
- **MLBL**: multimodal log-bilinear model, implementation of Kiros, Salakhutdinov, and Zemel (2014).
- **Image**: a system that for every new image, queries the set of training images for the most similar one, and returns a random description of that training example.
- **Keyword**: system that annotates every image with keywords that most probably describe it and then do a search query against all training data descriptions, returning the description that is closest (in terms of TF-IDF similarity) to the keywords.
- **Template**: system that uses templates inferred from dependency parses of the training data descriptions. A set of templates is discovered and a classifier that associates images with templates is trained.
- **SMT**: Ortiz et al. system first selects pairs of clipart objects that are important enough to be described by solving an integer linear programming problem, creates a “visual encoding” using visual dependency grammar and finally uses a phrase-based SMT engine to translate the latter to proper sentences.

Our system does not score as high as their machine translation system.

It is important to note that the descriptions given in the dataset, as well as those generated by the different systems are not “complete.” Each one of them describes a specific bit of information that is implied by the scene. Figure 5 demonstrates this. As such, the calculation of SMT evaluation scores with respect to a reference set is not necessarily the best mechanism to identify the correctness of a textual description. To demonstrate this point, we measure BLEU scores of one of the reference sentences while comparing it to the other references in the set. We did that for each of the eight batches of references available in the training set.

The average reference BLEU score is 24.1 and the average METEOR score is 20.0, a significant drop compared to the machine translation system of Ortiz, Wolff, and Lapata (2015). We concluded from this result that the SMT system is not “creatively” mapping the images to their corresponding descriptions. It relies heavily on the training set captions, and learns how to map images to sentences in a manner which does not generalize very well outside of the training set.

Another indication that our system creates a more diverse set of captions is that the number of unique captions it generates for the test set is significantly larger than that of the SMT system by Ortiz et al. The SMT system generates 359 unique captions (out of 2,004 instances in the test set), while CCA generates 496 captions, an increase of 38.1%.

To test this hypothesis about caption diversity, we conducted the following human experiment. We asked 12 subjects to rate the captions of 300 abstract scenes with a score

<table>
<thead>
<tr>
<th>system</th>
<th>BLEU</th>
<th>METEOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBL</td>
<td>7.3</td>
<td>17.7</td>
</tr>
<tr>
<td>MLBL</td>
<td>12.3</td>
<td>20.4</td>
</tr>
<tr>
<td>Image</td>
<td>12.8</td>
<td>21.7</td>
</tr>
<tr>
<td>Keyword</td>
<td>14.7</td>
<td>26.6</td>
</tr>
<tr>
<td>Template</td>
<td>40.3</td>
<td>30.4</td>
</tr>
<tr>
<td>SMT</td>
<td>43.7</td>
<td>35.6</td>
</tr>
<tr>
<td>CCA</td>
<td>26.1</td>
<td>25.6</td>
</tr>
</tbody>
</table>

Table 2: Scene description evaluation results on the test set, comparing the systems from Ortiz et al. to our CCA inference algorithm (the first six results are reported from the Ortiz et al. paper). The CCA result uses $m = 120$ and $\eta = 0.05$, tuned on the development set. See text for details about each of the first six baselines.
The ratings can be found here: http://cohort.inf.ed.ac.uk/canonical-correlation-inference.html.
References


Hodosh, M.; Young, P.; and Hockenmaier, J. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. JAIR 47:853–899.


