IMAGE-BASED BILINGUAL LEXICON INDUCTION

FOR LOW RESOURCE LANGUAGES

Brendan Daniel Callahan

A THESIS

in

Robotics

Presented to the Faculties of the University of Pennsylvania in Partial
Fulfillment of the Requirements for the Degree of Master of Science in Engineering

2017

_____
Chris Callison-Burch
Supervisor of Thesis

_____
Camillo J. Taylor
Graduate Group Chairperson

Abstract

Can images help us learn the translations of words? We introduce a new large-scale multilingual corpus of labeled images collected to facilitate research into learning translations through visual similarity. We collected 100 images for up to 10,000 words in each of 100 foreign languages, plus images for each of their translations into English. In addition to the images, we crawled the text from the web pages where each of the images appeared. Our dataset contains 35 million images and web pages, totaling 25 terabytes of data. As a basis for similarity, we use Scale Invariant Feature Transform (SIFT), color histograms and convolutional neural network (CNN) based features to compare images. Our generated bilingual lexicons show a significant increase in accuracy over previous work that used five high resource languages and concrete nouns. We see similar accuracies in many of our lower resource languages, and find that there is still plenty of signal when using abstract words, other parts of speech, and language-confident results only.

Acknowledgements

I would like to thank my thesis advisor Professor Chris Callison-Burch (CIS, University of Pennsylvania). I'd also like to thank co-advisor Professor Camillo Taylor (CIS, University of Pennsylvania). I would also like to thank John Hewitt (Student, CIS, University of Pennsylvania) for his assistance on Section 6.

Contents

1 Introduction

In machine translation (MT), translations are usually automatically acquired from bilingual sentence-aligned parallel texts (Brown et al., 1990). However, parallel texts do not exist for many domains and languages. Bilingual lexicon induction (BLI) is the task of learning translations without parallel corpora (Irvine and Callison-Burch, 2017). These bilingual lexicons are used for several tasks in the Natural Language Processing arena, including cross-language information retrieval (Lavrenko et al., 2002; Levow et al., 2005) and statistical machine translation (Och and Ney, 2003).

In order to find potential translation pairs using sets of monolingual text data in different languages, information that relates to the words must be used. Some researchers have used similar spellings across related languages to find potential translations (Koehn and Knight, 2002; Haghighi et al., 2008). Other researchers have exploited similar frequencies over time to induce translation pairs (Schafer and Yarowsky, 2002; Klementiev and Roth, 2006). Others still have tried using seed bilingual lexicons to help infer the context of unknown words, inducing additional bilingual pairings when a high contextual similarity exists (Fung and Yee, 1998; Rapp, 1999).

Given the volume of image data available on the web, images are another potential source of data for translation equivalence. Visual data can also be thought of as a language-independent way of representing information. Typically, the images appear on pages along with text in one or more languages. Facebook reported in 2008 that they had stored 10 billion photos, and currently they support 70 plus languages (Facebook, 2008). Google reported in 2010 that they had 10 billion images in their index, and they currently support over 100 languages (Google, 2010). When users upload images on Facebook, they will often provide some descriptive text in their native language. When Google indexes images, they typically pull in textual information from the websites that display the images, which can be in the form of image captions, targeted text snippets on the page, or the page text as a whole. This descriptive and identifying text can be thought of as a labeling scheme, where words and phrases are associated with images. Using these labels and exploiting the universality of images, we can find translations by

identifying images associated with words in different languages that have a high degree of visual similarity (Bergsma and Van Durme, 2011).

Several researchers have begun tapping this massive resource by deriving features from Google image search results and comparing them in order to induce bilingual lexicons. They evaluated lexicon induction with respect to English for each of Spanish, French, German, Italian and Dutch. They experienced substantial performance gains over the linguistic data alone when they combined these image-based features with linguistic data (Bergsma and Van Durme, 2011; Kiela et al., 2015). It has not been tried yet with very different languages like Arabic, Chinese or Hindi. Nor has it been tried with any lower resource languages like Indonesian, Turkish or Uzbek.

Though the previous corpora were available to us, we know of no corpus that contains a much larger set of words and languages. In order to assess how well image similarity can facilitate the task of learning translations on a much broader set of languages, our first goal was to establish a new dataset with many more words and languages.

How we differ from previous work

1. We work on a much larger scale (more words, more images)
2. We run experiments on many more languages
3. We try a mix of high and low resource languages
4. We use words other than just concrete nouns: other parts of speech and abstract concepts

The dataset we created is an improvement over previous datasets for this task, which were limited to a few high resource languages and to the translation of nouns. Our dataset contains images for 100 languages, and is not restricted by part of speech. We collected images using Google Image Search for up to 10,000 words in each of 100 foreign languages, and gathered images for a total of 263,098 English words. For each word, we collected up to 100 images. This allows us to test bilingual lexicon induction for low resource languages (where there is not enough sentence-aligned bilingual parallel data to train a machine translation system), and to investigate the performance of bilingual lexicon induction on different types of words (including verbs and adjectives as well as nouns). This paper introduces our methodology for creating the resource (which buffers against the criticism that Google may use bilingual dictionaries as part

of its image search).

We are interested in how our results compare to prior work on several fronts, in direct comparison with the high resource languages they used, when compared to more varied languages, and finally when compared to lower resource languages. Given a set of images associated with any foreign word, our task is to find the English word(s) that are most similar to it by comparing its images with the images of all English words. To accomplish this, we have created a pipeline that can take in images from two languages, compare the similarity of each set of images from both languages, and then rank the most likely translations in each case using Mean Reciprocal Ranking. Our pipeline is flexible and allows for experimentation with a variety of image features, including the SIFT, Histogram, and, Alexnet features. Our goal is to see how these techniques work when the languages start to differ more significantly than in previous tests.

We first re-ran the SIFT-based (Bergsma and Van Durme, 2011) and CNN-based (Kiela et al., 2015) experiments with our new corpus. We found that a noticeable improvement in our results over previous research for concrete words in five high resource languages. We also find a good amount of signal in our results on other high resource languages and even a few of our lower resource languages. We continue to find signal when we include abstract words and other parts of speech. In contrast to previous work, we perform two additional steps: filtering to use only language-confident images, and to skip same translation words. Our results decline somewhat after filtering, but much of the signal is still there.

2 Image Datasets

Corpus	Languages	Source	Num words	Nouns only?	Concrete only?	Used by
(Bergsma and Van Durme, 2011)	French, German, Spanish, Italian, Dutch	Google image search	500	yes	yes	(Bergsma and Van Durme, 2011; Kiela et al., 2015)
(Kiela et al., 2015; Vulic and Moens, 2013)	Spanish, Italian, Dutch	Wikipedia image captions	1,000	yes	mostly	(Kiela et al., 2015)

Table 1: Shows the breakdown of existing corpora used by prior work: what languages they cover, the source, how many words, the parts of speech, and the concreteness.

The most closely related work to ours is research into bilingual lexicon induction

using image similarity by Bergsma and Van Durme (2011) and Kiela et al. (2015). Their work differs from ours in that theirs focused more narrowly on the translation of concrete nouns for a limited number of high resource languages. Bergsma and Van Durme (2011) compiled datasets for Dutch, English, French, German, Italian, and Spanish by downloading 20 images for up to 500 concrete in each of the foreign languages, and 20,000 English words. Approximately 15% of the word pairs in their lexicon were same translation words.

Vulic and Moens (2013) generated another dataset, where they collected images for 1,000 words in Spanish, Italian, and Dutch, along with the English translations for each. This dataset consists of all nouns, but includes some abstract nouns along with the concrete nouns. We show a breakdown of the existing corpora in Table 1. Our corpus will allow researchers to explore image similarity for bilingual lexicon induction on a much wider range of languages and parts of speech, which is especially desirable given the potential utility of the method for improving MT between languages with little parallel text.

Recent research in the NLP and computer vision communities has been enabled by large collections of images associated with words or longer texts. Object recognition has seen dramatic gains in part due to the ImageNet database (Deng et al., 2009), which contains 500-1000 images associated with 80,000 synsets in WordNet. To our knowledge, there is no known equivalent for ImageNet in other languages. The ESP Game (Von Ahn and Dabbish, 2004) has also been used to label images with words, though like ImageNet it is only available in English.

Other NLP+Vision tasks that have been enabled by the availability of large datasets include caption generation for images, action recognition in videos, visual question answering, and others. Ferraro et al. (2015) surveys existing corpora that are used in vision and language research. Within the survey (Ferraro et al., 2015), we did find several additional corpora containing images labeled for English words with captions, including one data set of multilingual sentences describing short YouTube videos (Chen and Dolan, 2011). However, we were unable to find any image corpora that contained

multiple languages, with or without direct translations.

Kilgarriff (2007) aptly notes that it is difficult for academic researchers to compete with the resources of the tech giants with respect to availability of corpora. Our own attempts at finding existing corpora available to academics to broaden the set of languages and words we could work with also ended in failure, and therefore we opted to create our own.

3 Corpus creation

Our goal in establishing a new data set was to ensure we could use it for evaluating how well image similarity facilitates the task of learning translations. Rather than limiting our data to a few high resource languages and to the translation of nouns, we gathered images for many languages and did not filter out abstract words or restrict part of speech. We collected images from Google image search results for up to 10,000 words in each of 100 foreign languages. We also gathered images for a total of 263,098 English words [1]. For each word, we collected up to 100 images. This allows us to test bilingual lexicon induction for low resource languages (where there is not enough sentence-aligned bilingual parallel data to train a machine translation system), and to investigate the performance of bilingual lexicon induction on different types of words (including verbs and adjectives as well as nouns). A full breakdown of our dataset can be found in Table 22.

3.1 Example images

Figures 1, 2, 3, 4, 5 show the Google image search results for five known translation pairs. In each case, we show the top 50 words for the English word and its French translation. For the English word *cat* and the French word *chat*, we see a relatively consistent picture across the languages with lots of cat pictures, though in the French case, we see a few icons for the English word chat. In the case of the English word

[1] We split up the english superset into batches of 10,000

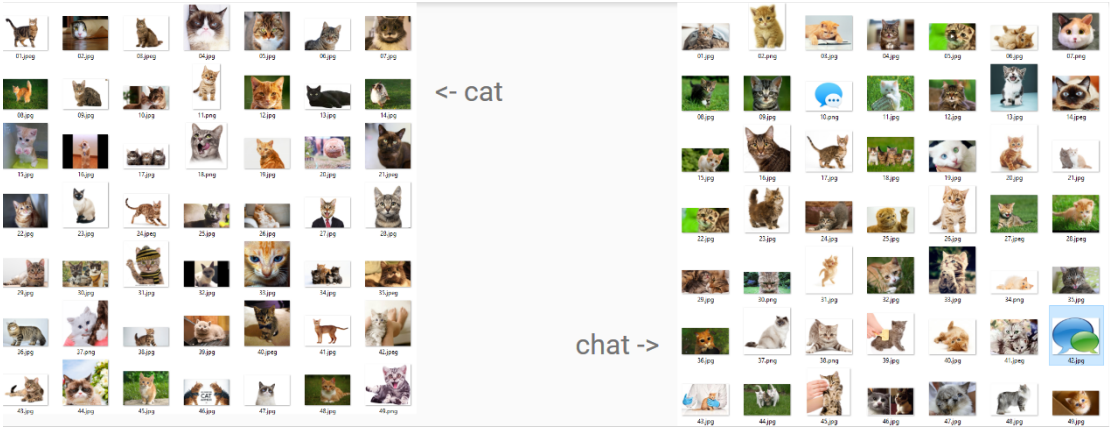

Figure 1: The top 50 images for the English word *cat* and the French word *chat*.

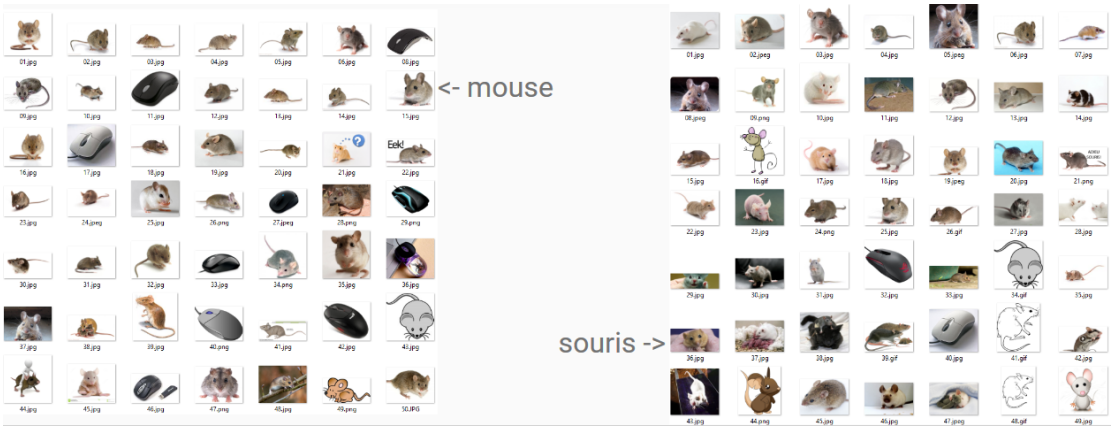

Figure 2: The top 50 images for the English word *mouse* and the French word *souris*.

mouse and the French word *souris*, we see also see consistency across the languages with lots of pictures of mice (the animal) as well as some pictures of computer mice. Computer mouse seems to be slightly more frequent an occurrence in the top results for English than for French. As we look at the English word *gold* and the French word *or*, we see mostly similar images across the languages with lots of pictures of gold bricks and other gold items. The one interesting case is that we see the words *gold* and *or* spelled out in their respective languages. For the English word *dam* and the French word *barrage*, we see with lots of pictures of dams in the result set for both languages. We do however start to see more blueprints of dams for *barrage* that are not coming up for *dam*. Finally, for the English word *metal* and the French word *métal* , we see a near identical set of images across the languages with lots of pictures of what looks like sheet metal.

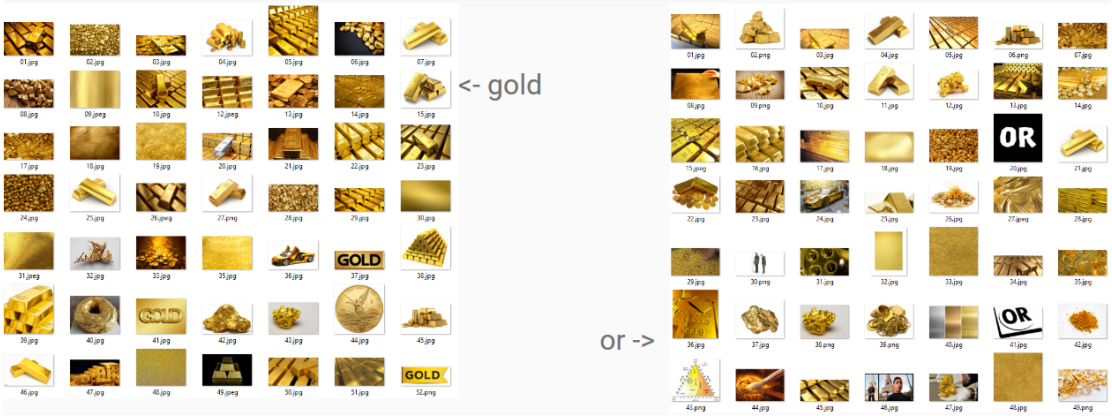

Figure 3: The top 50 images for the English word *gold* and the French word *or*.

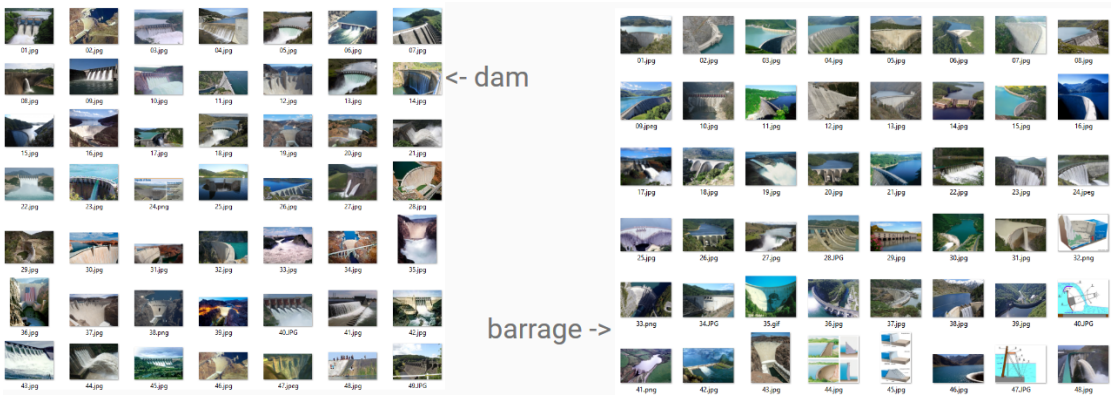

Figure 4: The top 50 images for the English word *dam* and the French word *barrage*.

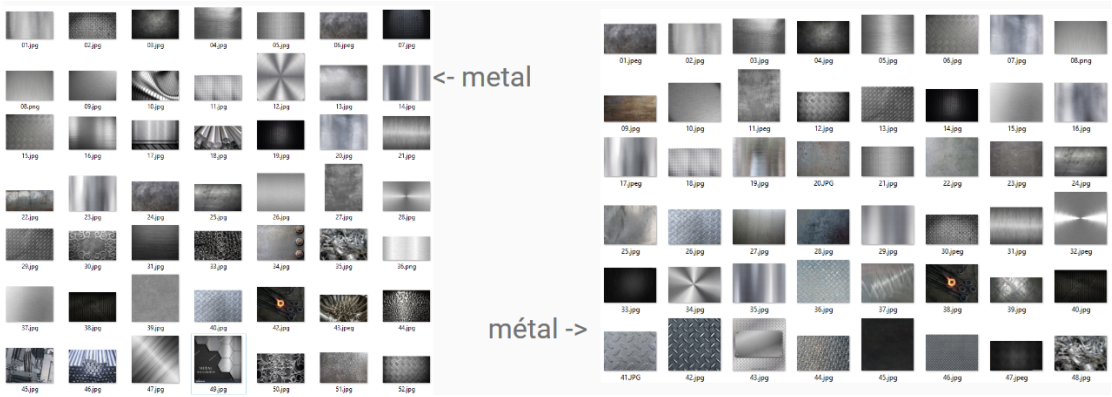

Figure 5: The top 50 images for the English word *metal* and the French word *métal* .

3.2 Process

To meet our goal of working with more languages and not restricting part of speech, we began looking at existing corpora with bilingual dictionaries in many languages. We found a set of bilingual dictionaries for 100 foreign languages, each paired with English translations (Pavlick et al., 2014). Pavlick et al. (2014) generated these bilingual dictionaries from a large-scale crowdsourcing experiment on mechanical turk. Only the most agreed upon translation results were used to create the dictionaries. Most of the dictionaries contain approximately 10,000 words, though due to the agreement threshold used in creating them, the exact number varies per language. As with Bergsma and Van Durme (2011)'s corpus, we find that approximately 15% of the words in our corpus for the five high resource languages described in 2 are same translation words.

The first decision we made was where to obtain the images from. We decided to use Google image search because we felt it would help us maximize the coverage of words in our bilingual dictionaries that produced image results. Google is the most popular search engine and it supports language-boosted search results for many of the languages we wanted to include in our corpus. Other researchers have attempted to evaluate Google-sourced datasets vs. others. One found that images from Google tend to provide higher quality representations than other sources (Bergsma and Goebel, 2011), while another found that Google-sourced datasets could compete with hand curated datasets (Fergus et al., 2005). Still another found that both Google and Bing were well suited to experiments with abstract words (Douwe Kiela and Clark, 2016). There are still valid concerns of doing research using Googleology (Kilgarriff, 2007), we felt that because our input dictionaries were created using mechanical turk and we attempt to filter out images that appear on pages not in the correct language (see Section 5.1), we are providing a reasonable simulation of a what monolingual image corpora might look like in various languages. We discuss this further in Section 4. We did not try using additional search engines due to time constraints.

The next thing we looked for was existing software that could help us create the corpus with minimal development. We researched a couple of open source solutions,

but none met our direct needs. To generate a corpus from simple bilingual dictionaries and Google image search, we decided to create our own purpose-driven solution. The overarching design behind our custom Google image scraper was to make it simple, flexible, and easy to maintain. With this design in place, we created a simple proof of concept so that we could evaluate initial results and see if our solution would help us create the corpus the way we wanted it. We decided to write our code in Python due to familiarity with the language and the ready availability of topically relevant and well maintained libraries.

It was also clear that we needed to find somewhere to run our scraper code, which would be downloading large quantities of data from a large number of web hosts. We were wary of running a massive scraping job on our personal or school resources, and decided that Amazon Web Services (AWS) ready availability of ephemeral resources was a natural fit. We had prior experience managing servers with AWS, and the ability to change the hostname generating web requests sealed the deal. AWS also offered a student credit of $100 that we were able to use to get our first experiments up and running with.

We wanted to make the process of starting a new scraping job for each language to be as easy as possible, especially given the number of languages we wanted to cover. We were able to achieve that simplicity by using several tools provided by AWSs Elastic Compute service (`EC2`). The first step was to create an instance based on the stock Ubuntu image provided by AWS. Once the instance was ready, we installed every package and library required to run our code. After making a few changes to the configuration details, we took an snapshot of the system volume for that instance, and fed that snapshot to the Amazon Machine Image (`AMI`) creation process (Figure 6). The only steps we left before images started downloading were those that were specific to the language being scraped. At this point, we could spin up as many instances as we needed with the exact same configuration, without having to repeat the setup steps.

This approach allowed us to create as many instances as AWS supports that we were willing to manage and pay for. Since scraping jobs are mostly network and storage

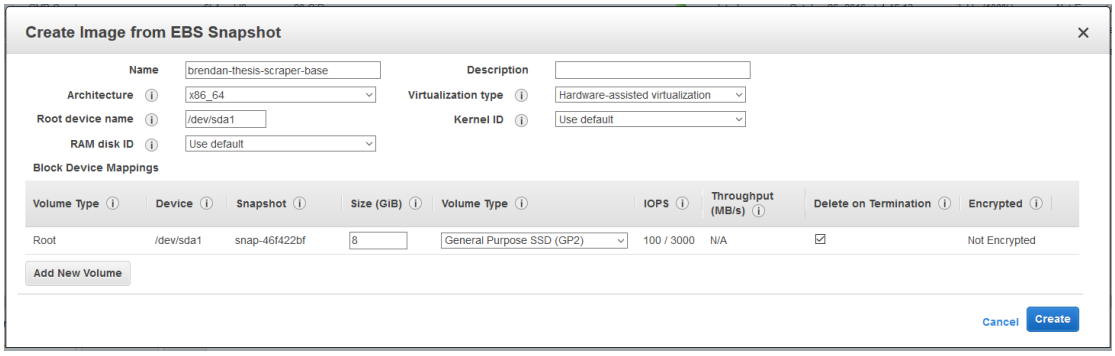

Figure 6: Screenshot when creating an `AMI` in AWS's console.

intensive, we were able to use a very cheap and small `t2.micro` instance to run our jobs. We chose to run our servers in the standard AWS region in Northern Virginia, which is relevant because that is where all of our web requests were originating from. The process for starting a scraping job took a couple of minutes and is outlined in the list below

1. Load an `EC2` instance based on the `AMI` (Figure 7)
2. Attach and mount a second storage volume to it
3. Run a few commands on the server to kick off the job.

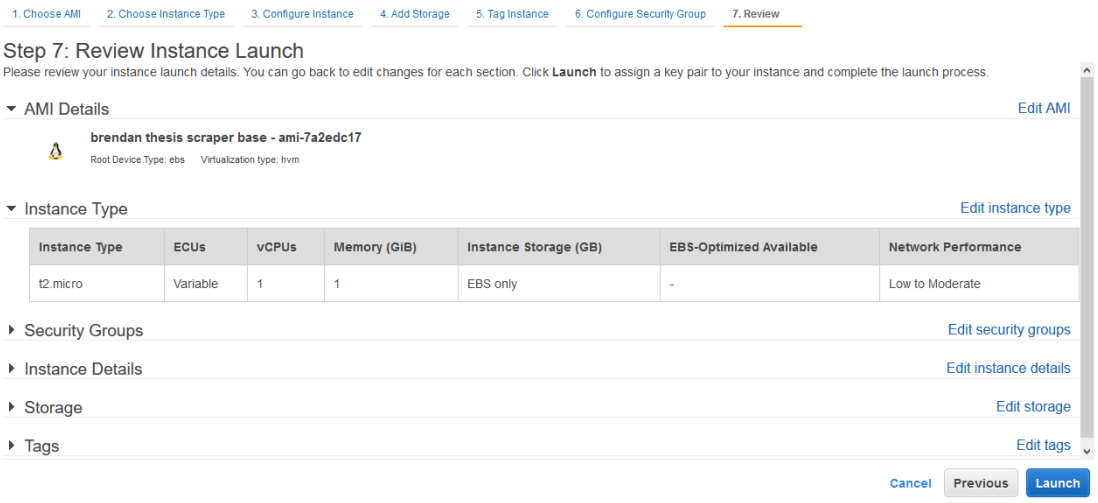

Figure 7: Screenshot when creating an `EC2` instance from our pre-made `AMI`.

We decided to interact with Google as if we were a normal user, rather than using their custom search API. While their API would have been preferable, we decided it was

not feasible for a corpus of our size due to request volume restrictions and cost considerations. We used several tools to ensure Google would treat our custom scraper running on a remote Ubuntu machine as a normal user. First, we turned to the `selenium` library (Selenium, 2006), which allows us to connect to a real web browser and programmatically control that browsers behavior. Figure 8 shows how simple it is to get started with it. We chose Firefox as the browser for mostly arbitrary reasons. In order to ensure our real browser worked in a headless environment, we used the `xvfb` package, which allows us to create a fake display that we can tell our browser to use. By starting up `xvfb` as a background process and configuring an environment variable, `selenium` and Firefox would know to use this fake display. Simpler prototypes before this (without a real browser) ran into various issues where Google searches required us to perform extra steps such as responding to Captchas. The solution was not without trickiness though, as the connection via `selenium`, `xvfb`, and Firefox would randomly fail and timeout. We also had to differentiate failure scenarios from when Google genuinely had 0 image results for a search term.

```
from selenium import webdriver
driver = webdriver.Firefox()
driver.implicitly_wait(10)  # wait for browser to
    load
search_url = 'https://www.google.com/search?#tbm=
   isch&start=0&hl=fr&lr=lang_fr&q=chat'
driver.get(search_url)
```

Figure 8: Simple example that uses `selenium` to fetch a Google image search results with French language tuning for the word *chat*

In order to interact with Google image search, we needed to create a base set of query parameters that could be easily extended on a per-language basis. The base parameters consisted of the flag to indicate we wanted to search for images and the flag to indicate we wanted to get the first 100 images for our query. Then, for each language, we decided to use Googles per-search language settings. They support two flags that relate to the language of the user making searches, `hl` (host language) [2] and `lr` (language

[2] "Explicitly setting this parameter improves the performance and the quality of your search results." (Google, 2017a)

restriction) [3]. The hl flag sets the user interface language and was available for approximately 85 languages in our corpus. The lr flag is an automatic language filter and was available for approximately 45 languages in our corpus. Unfortunately, documentation on the allowed values for this field is difficult to come across, but we found a list created by a third-party that helped us get a reasonable picture. We had to do minimal manual mapping of the source dictionaries with two letter language identifiers to the values for the hl and lr flags that only sometimes were identical.

Upon inspecting the search result html, we used XPath [4] to pick out the link elements that contained the source image information and a regular expression [5] to extract the links to the search images themselves. The raw link we extracted was a bit garbled though, and initially we had a successful download rate of 50%. Through experimentation, we determined that if we unquoted the raw link three times, we improved the rate of successful downloads to over 90%. In the node adjacent to the image links, we found that Google was dumping a bunch of JSON-based metadata about the source images. We augmented Googles metadata with the image link that we had extracted and parsed, the original filename, whether the download was successful, error information if it failed, and the filename used to store the result.

In order to facilitate downloading the source images, we needed a structure for storing the scraped images. We decided to use one outer folder for each language, one inner folder for each word's index in the dictionary, and then named the files based on their order of appearance in the search results, i.e. 01.jpg, 02.png, etc. We then set out to try downloading the source files using Python's `urllib` library. In our initial implementation to download the files, we experienced a large number of failures, due to a wide variance in size, format, and server host quality. We also found many files that had unknown or non-image extensions, which we worked around by inferring the extension from the content type when we went to download the file. There were also files that downloaded very slowly and/or timed out, which we worked around by setting

[3] "Language filters limit a search to pages in the specified languages. The Google Search Appliance has built-in language filters that detect the language of a query and return appropriate results." (Google, 2017b)

[4] //a[@class='rg_l']

[5] r'imgres\?imgurl=(?P<url>.*?)(&imgrefurl)'

a timeout value of 30 seconds. A timeout of 30 seconds was a compromise where we could end downloads that were definitely going to timeout and fail while still trying to download files that would pause here and there, but ultimately finish successfully. Download times are generally out of the developer's control, since they depend on the route used between our server and the hosting server, as well as the configuration of each of those servers controlled by third parties along the way. We found that download times for files varied widely from our servers hosted in Northern Virginia. Anecdotally, the vast majority of the files seemed to download very fast for web hosts in the USA and Europe, but very slowly for web hosts in China.

In downloading each of the source images serially, we estimated the time it would take to scrape just one of our 10,000 word dictionaries to be a couple of months. We felt this was completely intractable, and created a threaded implementation to download source images for each word that finished one 10,000 word dictionary in just under a week. This improvement was achieved with 6 threads running the downloads concurrently. We were wary of increasing the thread count too high, due to the possibility of flooding servers and being perceived as a denial of service (DoS) attack.

When testing the threaded implementation on our first dictionary, we received an email from the AWS EC2 Abuse team (Figure 9) that one of our instances "has been web-crawling at an excessive or disruptive rate" and that we should stop flooding those servers. The EC2 Abuse team had been forwarded server logs from one or more web host administrators that showed the IP address of our test server making multiple web requests with the `Python-urllib` user-agent header, which according to the logging software used by the administrators, constituted a DoS attack. We were fairly certain that the software they were using was looking for a specific list of user-agent headers in requests rather than looking at per-IP request rates. Immediately upon receiving this request, we shut down our scraping jobs and responded to the email that we were scraping for a school project and would temporarily stop until we finished improvements that addressed their concerns.

Regardless of the actual reason for being flagged, we still wanted to make a good

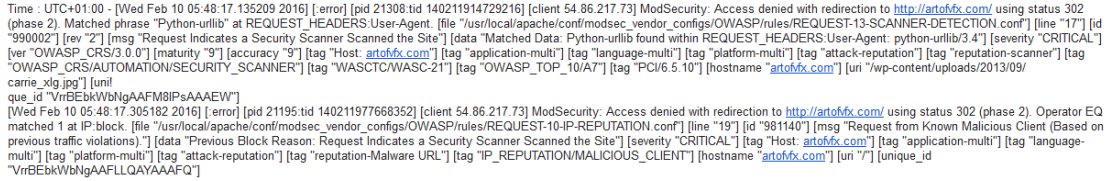

Figure 9: Snippet from AWS logs

faith attempt to reduce the potential for flooding any specific server with requests. In order to mitigate our risk, we decided to use an in memory cache to track the web hosts that we were downloading from. We felt it was overkill to limit downloads on any one host to one at a time, as it is common to make multiple requests to the same host while viewing a web site in your web browser. Therefore, we created a simple approach for throttling requests. If we had downloaded from a particular web host in the last 15 seconds [6], we paused initiating the next download against that host for 3 seconds [7]. We chose the Python library `beaker`, because it was recently maintained, and manages in memory caches reasonably well out of the box.

In order to address the likely root cause of being flagged by the EC2 Abuse team, we also needed to use something other than `Python-urllib` for our user-agent header. We decided to use a set of 120 commonly used user-agent headers on the web, which would allow us to blend right into their usual mix of requests. At minimum, we knew we'd no longer be flagged for using `urllib`'s default header value. The `EC2` Abuse team had suggested identifying ourselves explicitly in the user-agent header, but the prospect of receiving emails from less than expert system administrators about scraping image files that Google had already scraped anyways was unlikely to be constructive.

```
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:25.0) Gecko/20100101 Firefox/25.0
Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/535.19 (KHTML, like Gecko) Chrome/18.0.1025.45 Safari/535.19
Mozilla/5.0 (Android; Mobile; rv:29.0) Gecko/29.0 Firefox/29.0
Mozilla/5.0 (compatible; MSIE 10.0; Windows Phone 8.0; Trident/6.0; IEMobile/10.0; ARM; Touch)
Mozilla/5.0 (iPhone; CPU iPhone OS 7_0_6 like Mac OS X) AppleWebKit/537.51.1 (KHTML, like Gecko) Version
      /7.0 Mobile/11B651 Safari/9537.53
```

Figure 10: A couple example user agent headers that we used

Because our scraper jobs took multiple days, we ran them in named screen sessions. We ran `xvfb` in one screen session, and the scraper in the other. Both needed to be

[6] Based on experimentation
[7] Based on experimentation

running for the duration of each scraping job. Occasionally, we would see random failures to start our scraper due to Firefox being unable to connect to the `xvfb` display, but restarting `xvfb` would always fix the problem. Once the scraping job completed, we had occasional issues where search result requests would fail for one or more words. To address this problem, we created a flag for our scraper that allowed us to re-run the scraper only for words that had no image results. This second pass over the words was helpful for expanding the coverage of words in our dictionaries that had images downloaded for them.

We decided use AWS's Simple Cloud Storage Service (`S3`) service for the long term storage of our scraped corpora, given that we were already using AWS for the scraping jobs. For the first month of storage, we could store our data for approximately $0.03 per gigabyte, dropping to approximately $0.01 per gigabyte after the first month. Based on the first set of foreign words that we scraped (250GB), we estimated that 100 languages would conservatively take 25 terabytes to store. Our back of the envelope price estimate for the first month then came in around $750 per month, dropping to about $300 afterwards.

We created a single zipped file per language that we would upload to `S3` as the result. We found that transferring the images individually was incredibly slow, due to the overhead incurred with each request. We tried to optimize our file structure by using a single outer tar file that contained a `tar.gz` file for each word. This structure allowed us to extract the results for specific words, rather than always needing to extract the entire package just to access the images for a few words. We also decided to create a sample file for each language, consisting of 100 (1%) of the approximately 10,000 words in the corpus. This would allow somebody to download a sample file sized at approximately 2.5 GB instead of a package sized at 250 GB and evaluate if our files would work for them without needing to download an entire language first.

We wrote a separate Python script that creates both the package and the sample file, and then uploads them to `S3`. We also created the reverse script so that we could easily extract the results when we were done. In order to store the large amount of

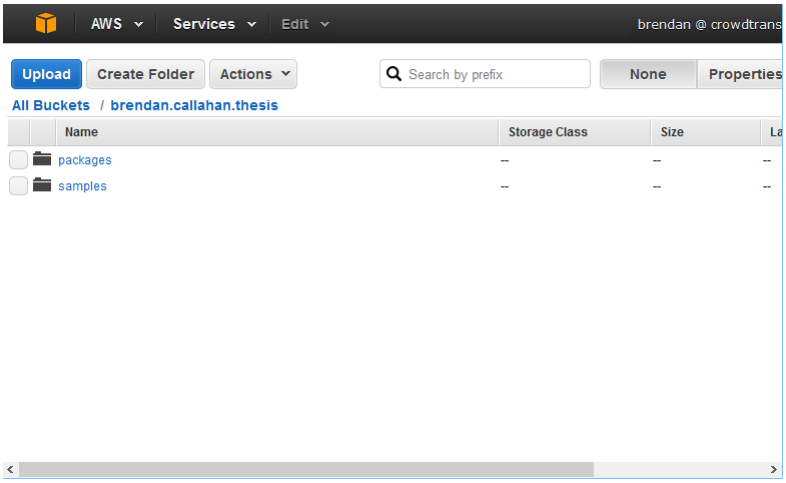

Figure 11: Top level AWS `S3` view of our bucket

intermediate data needed to generate our package file, we attached an additional storage volume to our instance, with approximately two times the amount of storage used by the downloaded files. Using cheaper, slower storage volumes on AWS, the approximate time to package and upload a 250GB package was 2 days. On a related note, we also needed a bucket key naming scheme to organize our results reasonably, we decided to create a single bucket with two top-level folders: packages and samples (Figure 11). Inside those folders was a file for each languages scraped results, and 27 distinct files for the English superset (Figures 12 and 13).

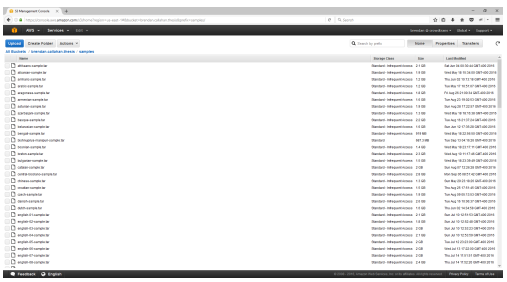

Figure 12: List of sample files as stored in our S3 bucket

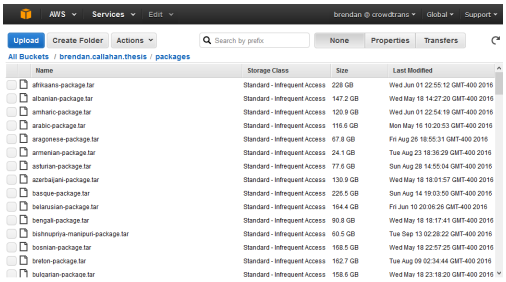

Figure 13: List of package files as stored in our S3 bucket

Given that all we required for running a scraping job was a newline separated text file with one word per line, we were able to run additional jobs as they came up. Part-way through the scraping process, we decided to create a dictionary for the Uighur language. In addition to this, based on the 2016 events surrounding an attempted coup in Turkey, we decided to scrape Turkish again after this news broke, as we thought it

might be an interesting comparison point for words that might relate to the coup and how their images changed after the event. The final amount of data that we scraped with 100 languages, plus 25 sections of the English superset, Uighur, and Turkish a second time was approximately 21 terabytes.

We have open sourced the code we used in a repository named multilingual-google-image-scraper on Github[8]. The code is generalizable to any input dictionary and we provided instructions on how to run it on a variety of platforms.

Dependencies:

1. AMI: ubuntu-trusty-14.04-amd64-server-20160114.5
2. Additional aptitude packages: python-virtualenv libxslt-dev libxml2-dev python-dev python3-dev zlib1g-dev git unzip xvfb firefox
3. Python version: 3.4.5 with virtualenv
4. Pip libraries: beaker awscli selenium

4 Is this Googleology?

One concern of ours has been: what if Google is just using a bilingual dictionary under the hood and serving up images from English webpages for foreign language searches?

There are potential downsides to using Google image search in general, and the `hl` and `lr` language customization flags specifically. We don't know what Google is doing behind the scenes when running searches, and it is possible that they are translating queries into English (or other high resource languages) and returning images associated with the translated queries (Kilgarriff, 2007), rather than in the actual language we are trying to generate images for, in an attempt to improve the relevancy of their search results. It is even conceivable that in using image data from Google that we are effectively reverse engineering internal bilingual dictionaries that they have built in a variety of languages.

To counter this, we take steps to filter out images that did not appear on pages written in the language that we were gathering images for. We do so by performing language identification on the text that is associated with the images. First we create a

[8] https://github.com/brendandc/multilingual-google-image-scraper

parallel corpora of the text from the pages the images appeared, as recorded in Google Image Searchs metadata. Then for the text of each image, we use the Chrome Language Detector 2 (Sites, 2013) to detect that pages native language. Then, we try evaluating our corpus again by removing those images where the language detected on the page differed from the language of the set of words they are assigned to.

5 Complementary Text Corpus[9]

Source Language	Arabic		Dutch		French	
Detected Language	Arabic	.5543	Dutch	.5139	French	.5822
	English	.4189	English	.4539	English	.4038
	Persian	.0051	German	.0078	Spanish	.0022
	French	.0020	French	.0056	Norwegian	.0017
	Norwegian	.0015	Norwegian	.0022	German	.0013
Source Language	German		Italian		Spanish	
Detected Language	German	.5946	Italian	.5856	Spanish	.6144
	English	.3847	English	.3797	English	.3609
	Dutch	.0038	Spanish	.0109	Portuguese	.0115
	French	.0032	Portuguese	.0063	Galician	.0017
	Norwegian	.0019	French	.0036	Italian	.0015

Table 2: The top-5 most common languages detected in individual pages for each of 6 high-resource languages. With each language is the fraction of web pages represented by that language.

A heuristic used with great success by search engines is that the words used on a webpage containing an image are likely to be related to that image. By extracting the text of the webpages that displayed the files in our image corpus, we are able to accomplish dual goals of ensuring the text is in the language of interest and enhancing the dataset by providing a "comparable corpus." A comparable corpus is a multilingual dataset with some noisy signal of translation equivalence. In our case, words extracted from webpages of similar images are likely to be topically similar. Because the image similarities are language-independent, we get a noisy multilingual signal.

Due to the vagaries of the internet, we were able to extract text for approximately 78% of the images in the corpus. Tables 2 and 3 show the top-5 most common languages

[9] John Hewitt (Student, CIS, University of Pennsylvania) did most of the work in this section.

detected in the text corpus, along with the fraction of web pages represented by that language, for 6 high- and low-resource languages of interest, respectively. Each web page does not necessarily correspond to a single image in the image corpus; any web page could be shared by many images.

Source Language	Bengali		Cebuano		Indonesian	
Detected Language	Bengali	.7256	English	.8318	English	.4972
	English	.2300	Spanish	.0401	Indonesian	.4667
	Russian	.0160	Tagalog	.0264	Malay	.0114
	Tajik	.0083	Cebuano	.0227	Turkish	.0034
	Bulgarian	.0073	French	.0129	German	.0026
Source Language	Turkish		Uighur		Uzbek	
Detected Language	Turkish	.6772	Uighur	.6609	English	.6035
	English	.3077	English	.1140	Uzbek	.1882
	Spanish	.0013	Inupiaq	.0778	Russian	.1169
	Indonesian	.0011	Arabic	.0291	Turkish	.0290
	German	.0011	Persian	.0290	Azerbaijani	.0128

Table 3: The top-5 most common languages detected in individual pages for each of 6 low-resource languages. With each language is the percent of web pages represented by that language. Note that we used a separate, unpublished language detection system for Uighur because CLD2 does not support Uighur detection.

The percentage of web pages written in the language of interest varied greatly from language to language, but was typically between 50% and 60% for high-resouce languages. Qualitatively, many pages were from YouTube or other English-speaking sites that happened to rank highly on foreign-language image searches. This motivates the necessity of filtering images used in the bilingual lexicon induction task to just those coming from in-language web pages.

Figures 14 and 15 show examples of text extracted from a page retrieved from the image search metadata. The text is paired with the image that appeared on its web page, as well as the Indonesian word used in the image search.

5.1 Language-confidence

We used a heuristic for language-confidence such that if an expected language showed up in the top-3 most likely languages as output by our language detection system on a web page, images on that page were kept. This relatively lenient heuristic is well-

Referring url
http://thayyiba.com/2015/12/15/2799/keistimewaan-memelihara-kucing/

Indonesian (extracted from web page)
Kucing merupakan salah satu jenis hewan yang banyak dipelihara oleh kebanyakan orang. Wajahnya yang lucu dan imut menjadika n kucing adalah teman bermain yang menggemaskan. Belum lagi tingkah polah dari kucing yang kerap manja serta menarik perhatian membuat kita betah berlama-lama bermain dengan kucing.

Translation (done for illustrative purposes)
Cats are one of the animals that many people keep. Their funny and cute faces make cats adorable playmates. Not to mention their behaviors are often affectionate and done to attract attention, which makes us enjoy spending a lot of time playing with cats.

Figure 14: Example text extracted from a web page corresponding to an image found for the Indonesian word *kucing* (cat), and the same text manually translated to English.

motivated because of the nature of automatically-scraped text from the web. English text is pervasive on the internet, even when the primary language of content of the page is not English. Further, many pages with our images have small amounts of text.

In all cases, we jointly attempt to detect all languages on a given page using the CLD2 library (Sites, 2013). When the language of interest shows up in the top 3 guesses, we have reasonable evidence that some of the webpage's text is in that language, even if there's also a substantial amount of English or some other language. Any multilingual web pages are valid for our purposes and should be kept. Table 4 shows the percentage of pages in the expected language for an 11 language subset of our corpus.

Referring url
http://www.bbc.com/indonesia/vert$_e$arth/2015/09/150911$_v$ert$_e$arth$_k$ucingstres
Indonesian (extracted from web page)
Apa yang terjadi jika kucing stres ?

Translation (done for illustrative purposes)
What happens if the cat is stressed?

Figure 15: Example text extracted from a web page corresponding to an image found for the Indonesian word *kucing* (cat), and the same text manually translated to English.

6 Concreteness of words

In order to measure the concreteness of words, we use two sets of English words labeled with concreteness ratings: the University of South Florida norms (USF) (Nelson et al., 2004) and the University of Ghent dataset (Ghent) (Brysbaert et al., 2014).

The USF dataset uses a scale of 1 to 7 to represent concreteness, where 1 is the most abstract, and 7 is the most concrete. They borrowed most of the values from prior research. The Ghent dataset uses a scale of 1 to 5 to represent concreteness, where 1 is the most abstract, and 5 is the most concrete. They created their results via a large-scale internet crowdsourcing experiment in the form of a norming study.

The USF dataset has concreteness ratings for 3,260 words, while the Ghent dataset has concreteness ratings for 39,954 words. Table 5 shows a breakdown of word counts and percent coverage for the English words in their corpora and our corpus.

language	percent in expected language
Bengali	76%
Turkish	69%
Spanish	64%
Arabic	64%
German	64%
Italian	62%
French	61%
Dutch	54%
Indonesian	50%
Uzbek	23%
Indonesian	6%

Table 4: Language identification statistics for 11 languages in our corpus using CLD2

	USF	Ghent
English words in their corpus	3,260	39,954
Words covered in our corpus	3,125	19,871
Percentage of words they have concreteness scores for that are in our corpus	96%	50%
Percentage of our corpus they have concreteness scores for	1%	8%

Table 5: Summary statistics for the concreteness corpora relative to the words in our corpus

Concrete words

USF	***Ghent***
1. ambulance - 7.0	1. tulip - 5.0
2. arrow - 7.0	2. telescope - 5.0
3. elephant - 7.0	3. elephant - 5.0
4. strawberry - 7.0	4. strawberry - 5.0
5. flask - 7.0	5. bedsheet - 5.0

Figure 16: Some examples of very concrete words in (Nelson et al., 2004) and (Brysbaert et al., 2014).

Figure 16 shows some very concrete word examples, while Figures 17 and 18 show the top 50 image results in our corpus for some of those same very concrete words.

For the top 50 images for the word *elephant*, we saw a relatively consistent situation where all of the images are photographs or drawings of elephants, with the head and trunk visible, sometimes showing one elephant and other times showing multiple elephants. For the top 50 images for *strawberry*, we see a similar situation, where everything is either a clear picture or drawing of a strawberry or strawberries, with the body and stem clearly identifiable in all cases.

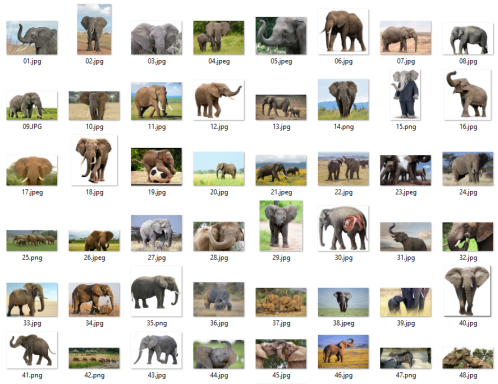

Figure 17: The top 50 image results for the concrete word *elephant* that has the maximum concreteness rating in both corpora.

Figure 18: The top 50 image results for the concrete word *strawberry* that has the maximum concreteness rating in both corpora.

Abstract words

USF	***Ghent***
1. hope - 1.18	1. essentialness - 1.04
2. thought - 1.28	2. eh - 1.04
3. moral - 1.39	3. spirituality - 1.07
4. morals - 1.39	4. although - 1.07
5. virtue - 1.46	5. possibility - 1.33

Figure 19: Some examples of very abstract words in (Nelson et al., 2004) and (Brysbaert et al., 2014).

Figure 19 shows some very abstract word examples, while Figures 20 and 21 show the top 50 image results in our corpus for some of those same very abstract words.

In the top 50 images for the word *hope*, it is difficult to pin down a consistent theme to the images. We see the word itself spelled out in many fonts and with many backgrounds. We can see some themes like bright rays of sunlight, but they are still on varied backgrounds and other identifiable objects are present. For the word *possibility*, we see another situation where we would struggle to identify a consistent theme to the images. We do see some cases where the word itself is written (typically with some other prose), and a few places where doors are opening to signify possibilities, but there is a lot of variance in the set.

Figure 20: The top 50 image results for the abstract word *hope* that rates at 1.25 in the Ghent corpus and 1.18 in the USF corpus.

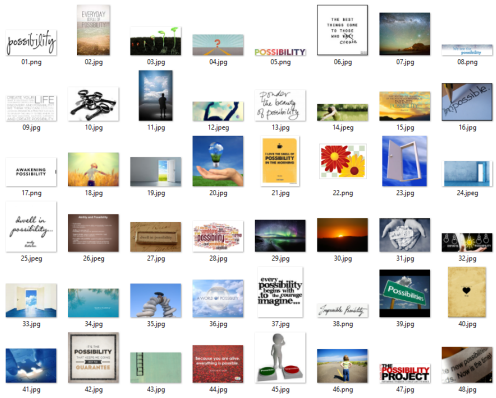

Figure 21: The top 50 image results for the abstract word *possibility* that rates at 1.33 in the Ghent corpus and 1.52 in the USF corpus.

7 Visual features

7.1 SIFT

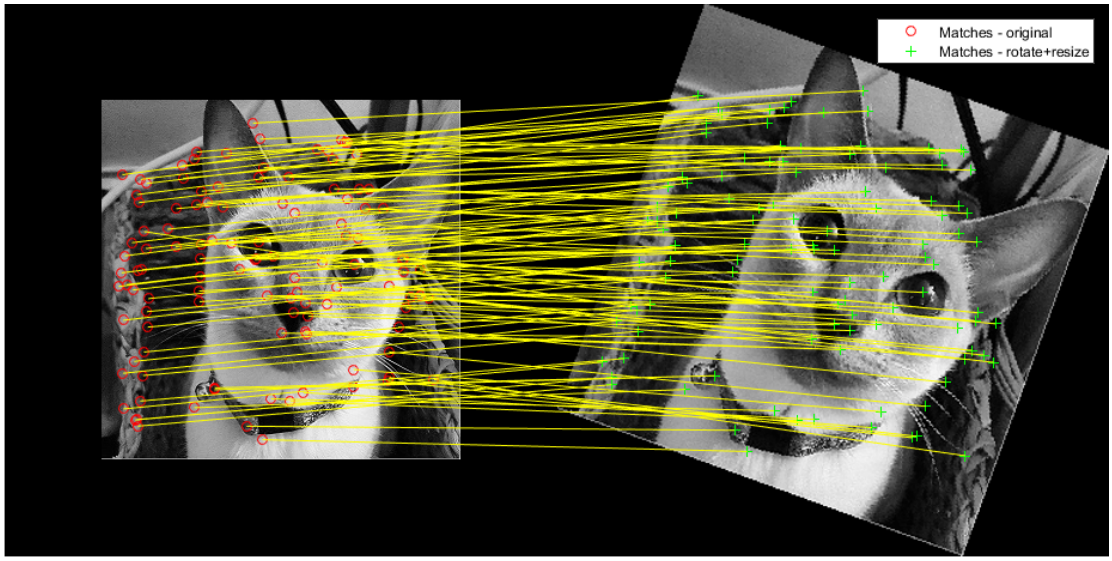

Figure 22: Original picture on the left with matching features highlighted for a rotated and scaled up version of itself.

Scale-invariant feature transform (SIFT) (Lowe, 2004), is an algorithm for transforming image content into local feature coordinates that are invariant to translation, rotation, and scale. These features (keypoints) are also robust enough to work through changes in illumination, noise, and distortion. These SIFT keypoints can then be generated for different images, and we can use features that appear in both images to help

```
930        128
11  4  0  0  0  0  0  6  127  12  0  0  0  2  2  141  24  0  0  7  78
     13  4  64  0  0  0  9  57  0  0  0  44  23  0  0  0  10  29  11
    141  86  1  5  6  7  19  66  68  7  1  82  141  21  2  13  0  0
    0  24  103  1  0  0  15  1  0  0  0  53  141  24  141  9  4  47
    26  23  129  131  32  5  5  141  93  0  0  5  0  0  0  29  11  0
     0  0  12  1  0  0  0  3  17  7  1  0  0  37  80  6  27  10  0  0
    0  44  83  7  0  0  0  0  0  1  0  0  0  0
```

Figure 23: First two lines of an example SIFT keypoint representation for an image. 930 represents the number of SIFT keypoints in the file, 128 is the number of dimensions in the vectors. Note: that 128 is the number of dimensions returned by the SIFT keypoint algorithm.

gauge similarity between those images. Figure 22 shows SIFT features being matched from an original image and a version that is rotated and scaled up.

The number of SIFT keypoints generated for each image file varies widely, but the number of dimensions per feature is constant (128). The set of SIFT keypoints can be thought of as a bag of visual words. Figure 24 shows five cat pictures with the strongest SIFT keypoints in each highlighted. We show an example raw SIFT keypoint vector in Figure 23.

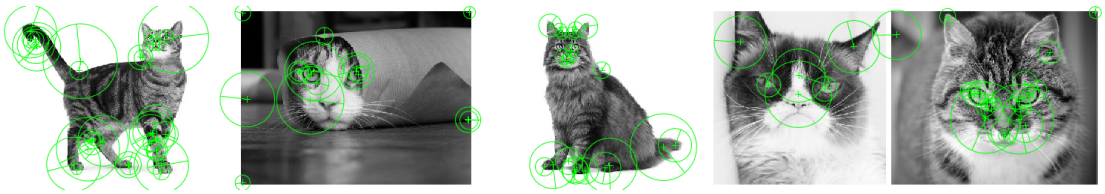

Figure 24: Five individual cat pictures with the strongest features highlighted and visualized.

(Bergsma and Van Durme, 2011) adapted the library written by (Lowe, 2004) to generate and store SIFT keypoints for candidate images with a c++ library of their own. We have in turn adapted their library to generate keypoints for a larger and less standardized corpus of images in Python.

In order to make meaningful comparisons, we need to group similar SIFT keypoints together. In order to convert the sift keypoints (represented as 128-dimensional vectors) into an easily comparable bag of visual words, we cluster the keypoints (Bergsma and Van Durme, 2011) using the k-means clustering algorithm. We used the mini batch k-

means (Sculley, 2010) approximation to speed up computation and randomly sampled 20% of the SIFT keypoints to avoid memory constraints. Following the lead of previous researchers (Bergsma and Van Durme, 2011), we configure k-means to output 100 clusters, though experimenting with different values would surely be worthwhile.

```
1: 40
2: 30
3: 30
```

Figure 25: SIFT cluster occurrence snippet

```
1: 0.4
2: 0.3
3: 0.3
```

Figure 26: Normalized SIFT cluster count snippet

We then assigned each SIFT keypoint to the cluster nearest to it, counting the number of keypoints in each cluster per image file. Finally, we normalize the per-file counts of each of the 100 dimensions so that the relative number of SIFT keypoints between images is irrelevant. We use these normalized cluster counts as the input representation for computing image similarity (Bergsma and Van Durme, 2011). We show examples of the raw and normalized cluster counts in Figures 25, 26.

7.2 Color histogram

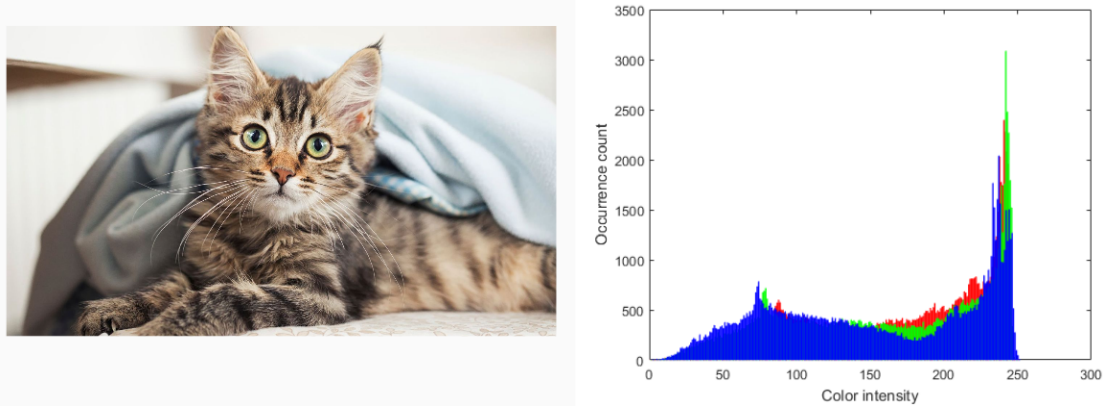

Figure 27: Picture of a cat with the graph of RGB color intensities (0-255) and occurrence counts. Note that in this example the colors skew towards higher intensities and therefore lighter colors.

We also create features using color histograms (Deselaers et al., 2008) for each of the images in our corpus in parallel with the SIFT features. We abbreviate color

histogram as HIST. We first generate a raw histogram of the RGB values for each pixel in the image using the ImageMagick utility. (ImageMagick Studio, 2008).

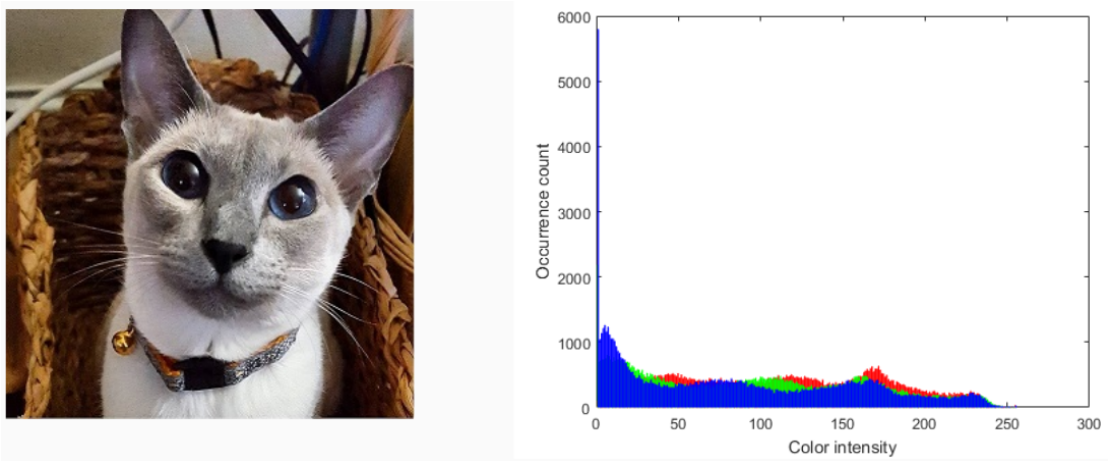

Figure 28: Picture of a cat with the graph of RGB color intensities (0-255) and occurrence counts. Note that in this example the colors skew towards lower intensities and therefore darker colors.

We then use the first hexadecimal digit of each R, G, and B value for each pixel to create a slightly generalized count of the number of pixels that fall into each of the 4,096 possible combinations (feature dimensions) of the three hexadecimal digits.

```
461: (   1,   9,   1) #010901  srgb(1,9,1)
319: (   4,  20,  28) #04141C  srgb(4,20,28)
891: (   4,  23,  35) #041723  srgb(4,23,35)
```

Figure 29: Raw histogram result snippet

```
F02:  30
EA3:  20
6A1:  50
```

Figure 30: Histogram occurrence snippet

```
F02:  0.3
EA3:  0.2
6A1:  0.5
```

Figure 31: Normalized histogram count snippet

We then normalize these feature counts so that the relative number of pixels between images is irrelevant. The normalized counts that result from this process are used as the input representation for computing image similarity (Bergsma and Van Durme, 2011). We show examples of the three steps in the process in Figures 29, 30, 31.

7.3 Convolutional neural network

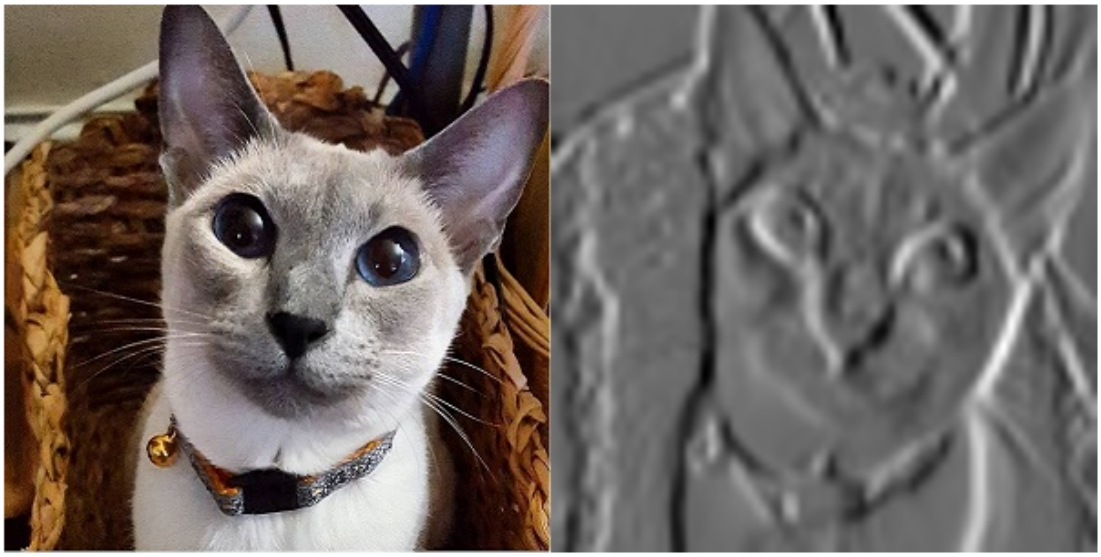

Figure 32: Strongly activating feature in first convolutional layer compared to the source image. We can still make out the overall outline of the cat fairly easily here.

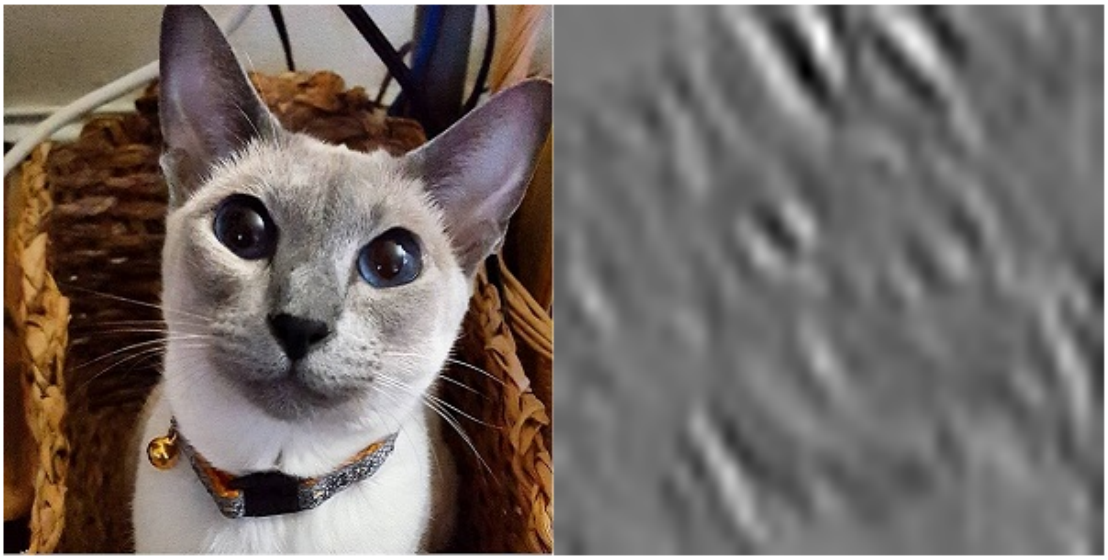

Figure 33: Strongly activating feature in second convolutional layer compared to the source image. The outline of the cat is much more difficult to see.

In addition to evaluating the corpus using SIFT features, we also try to use image features that are created with deep convolutional neural networks (CNNs). CNNs are a type of feed forward neural network that attempt to model visual perception in animals. They include a set of convolutional layers, within which sets of filters (kernels) are applied to (convolved with) the two-dimensional image data. CNNs provide top perfor-

mance for tasks like object recognition and have advanced the field of Computer Vision by a similar amount to when SIFT features were first introduced (Sharif Razavian et al., 2014). See Figures 32, 33, 34, 35, and 36 to compare source image data with strongly activating convolutional layer features at different levels.

Figure 34: Strongly activating feature in third convolutional layer compared to the source image. The outline of the cat is difficult to see and we start to see areas of interest around the eyes and nose.

Figure 35: Strongly activating feature in fourth convolutional layer compared to the source image. The outline of the cat is mostly gone here, but we see what looks like a cat mask, with the eyes clearly identifiable.

Figure 36: Strongly activating feature in fifth convolutional layer compared to the source image. We can't see the overall outline of the cat here, but can still identify what look to be the eyes and the bell.

We choose to start by replicating past work where image features extracted from the `fc7` layer of a trained AlexNet (Kiela et al., 2015) were used. The AlexNet neural network (Krizhevsky et al., 2012) starts with five convolutional layers, continues with two fully connected layers (including the `fc7` layer), before feeding into the softmax regression algorithm. It has been pre-trained with the ImageNet (Deng et al., 2009) classification task using the Caffe deep learning framework (Jia et al., 2014). Douwe Kiela and Clark (2016) found that circa-2012 AlexNet CNNs perform comparably to VGGNet (Simonyan and Zisserman, 2014) and GoogLeNet (Szegedy et al., 2015) CNNs in multi-modal tasks.

We use the Python extensions for Caffe (Jia et al., 2014) via an adaptation of the mmfeat library (Kiela, 2016) for working with AlexNet features. By taking the output of the `fc7` layer as our featurized image, we can directly compare the features for two images using cosine similarity. We can skip the k-means clustering step that we needed to do for the SIFT features because the CNN features can be directly compared.

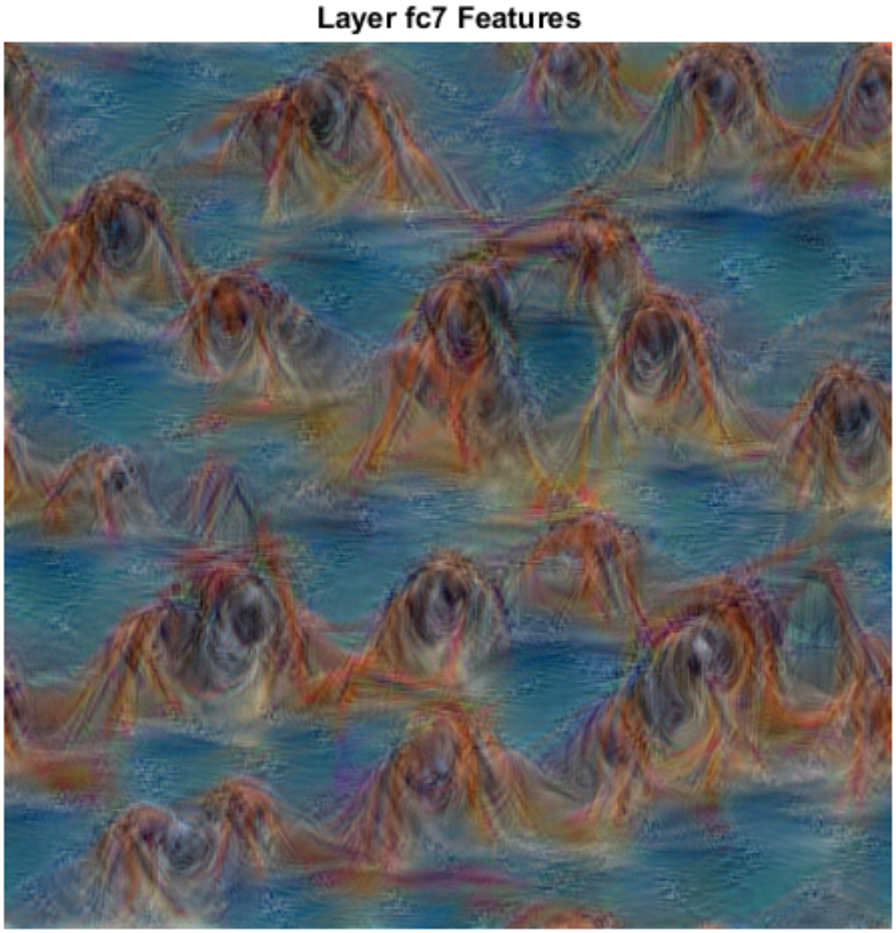

Figure 37: Synthesized deep dream image strongly activating the Siamese cat feature in the `fc7` layer.

7.4 Image similarity

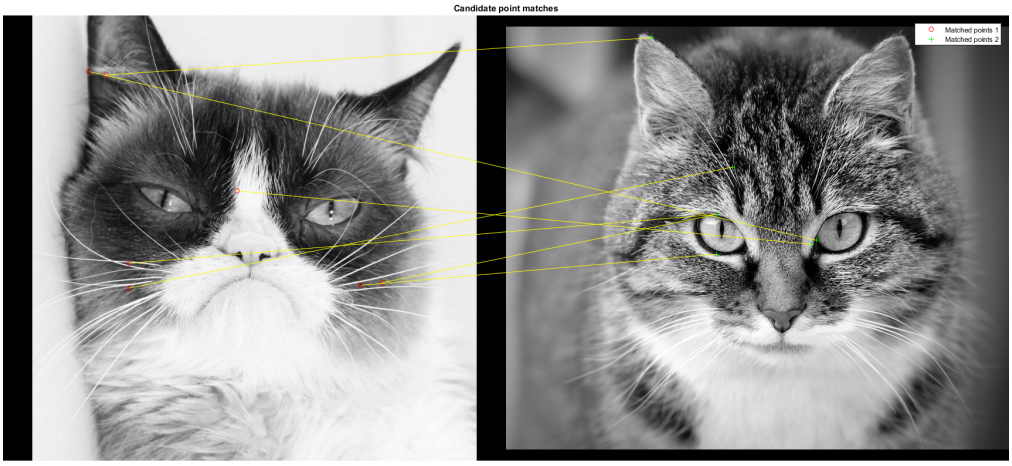

Figure 38: Matching SIFT features between two different cat pictures.

In order to compute similarity for any image pairing, we take the feature vectors for each image. In the SIFT+HIST case (shown in Figure 38), we take each images normalized SIFT cluster counts and vectors of histogram feature counts, respectively, and compare them using cosine similarity. In the AlexNet scenario, we can directly compare the 4,096 dimension `fc7` layer using cosine similarity. In the SIFT+HIST case, we arbitrarily weight the SIFT features 2 times more than the HIST features.

AVGMAX

Cosine similarity is a commonly used distance metric for vector-space models of language (Turney and Pantel, 2010). We take these cosine similarity scores and use a linear combination of the results for SIFT cluster counts and histogram features to create a weighted similarity score between the two vectors (Bergsma and Van Durme, 2011).

The next step is to get similarity scores for pairs of words or phrases. We start by taking the maximum cosine similarity score for each of the foreign words N images with each of the M images for the English word. We then take the average of those maximum cosine similarity results. We refer to this as the `AVGMAX` method. w_f and w_e are the sets of images for the English and foreign word being compared, while i_f and i_e refer to the member images of each set. Figures 39, 40, and 41 show an example

walkthrough of the AVGMAX method with one foreign word vs. three English words.

$$\text{AvgMax}(w_f, w_e) = \frac{1}{|w_f|} \sum_{i_f \in w_f} \max_{i_e \in w_e} (cosine(i_f, i_e))$$

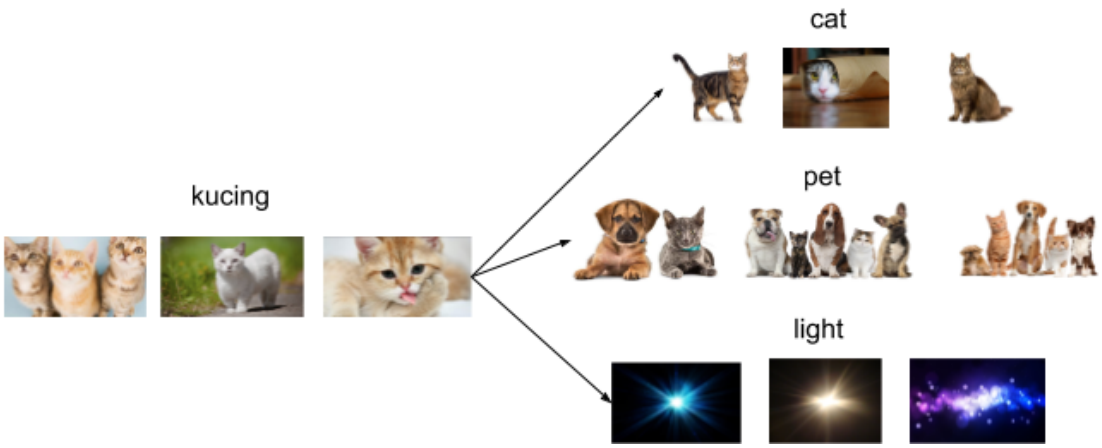

Figure 39: Step 1 AVGMAX process, comparing the Indonesian word *kucing* to three English words: *cat*, *pet*, and *light*

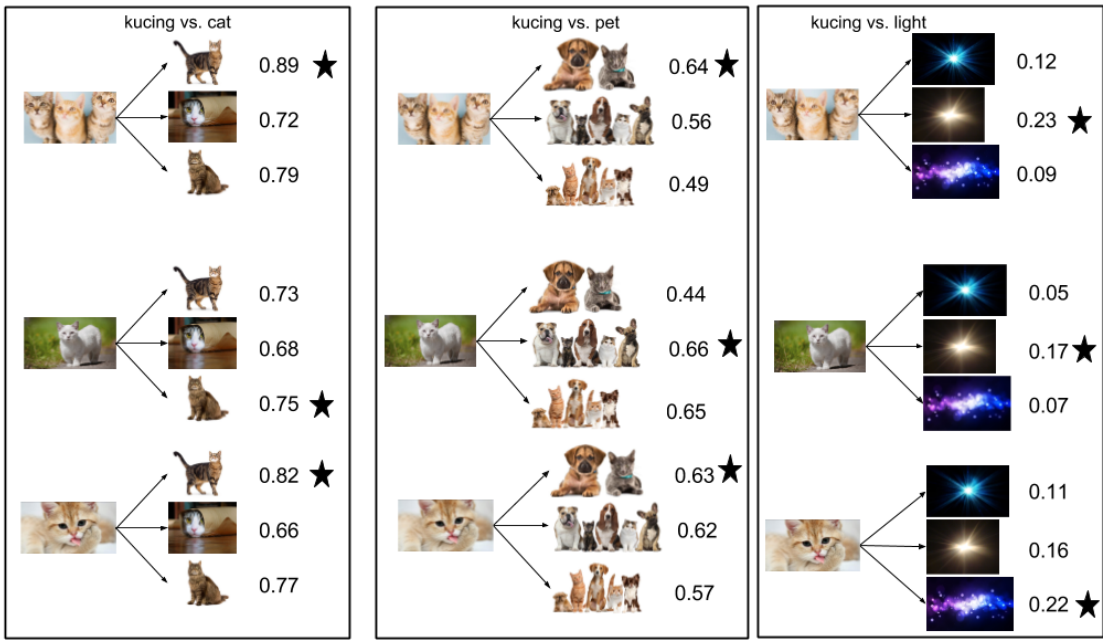

Figure 40: Step 2 AVGMAX process, comparing each image for the Indonesian word *kucing* to each image in each of the three English words: *cat*, *pet*, and *light*. The result at this step is the maximum cosine similarity for each image in *kucing* vs. the three options in each English word.

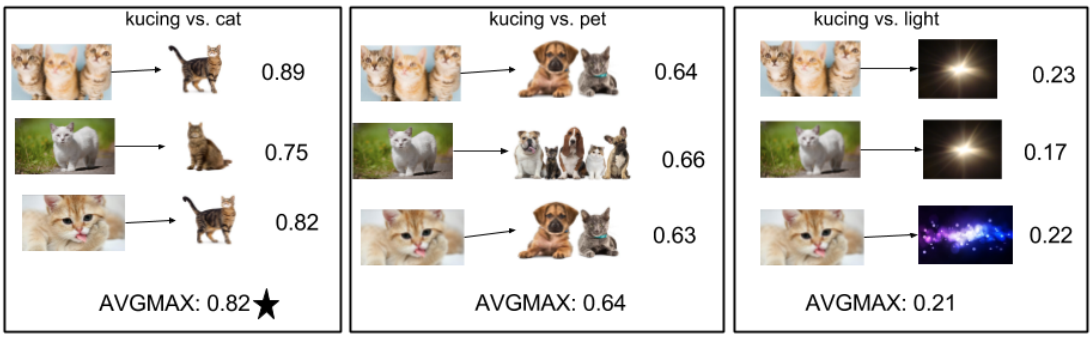

Figure 41: Step 3 AVGMAX process, averaging the max cosine similarity of each image for the Indonesian word *kucing* vs. the possible options in each of the three English words: *cat*, *pet*, and *light*. The result at this step is the AVGMAX value that feeds into ranking for MRR and P@N.

MRR

To evaluate a foreign language pair against English, we use the `AVGMAX` method to get a score between each possible word pairing. We have the known translations for each language from our bilingual dictionary corpus (Pavlick et al., 2014). We then rank the possible English results for each foreign word based on these scores and use Mean Reciprocal Rank (MRR) to evaluate the per-language results.

$$\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\text{rank}_i} \tag{1}$$

We also evaluate based on the precision of the correct translation being the top option, or part of the top 5 or 20 options, which we call the Top-N accuracy metric and is otherwise known as precision at N (P@N) (Gaussier et al., 2004; Tamura et al., 2012; Vulic and Moens, 2013).

8 Image dispersion

To help us better understand our dataset, we are also interested in the similarity between the images in the image sets for each of the words in our corpus. Kiela et al. (2014) proposed using Image Dispersion scores to measure this intra-word similarity of images.

For each word in our corpus, we have compared all of the images for that word to each other and outputted a single value that represents how similar the images are to each other. We can calculate the Image Dispersion score for a word by computing the average pairwise cosine distance between all of the image representations for any given word. Kiela et al. (2014) chose average pairwise distance to emphasize the total variation.

$$\text{d}(w) = \frac{1}{2n(n-1)}\sum_{i\leq j\leq n} 1 - \frac{\overrightarrow{w_i}\cdot\overrightarrow{w}_j}{|\overrightarrow{w_i}||\overrightarrow{w_j}|}$$

We first tried generating image dispersion scores using dense SIFT features (Bosch et al., 2007). We also tried generating dispersion scores with Pyramid Histogram of

Visual Word (PHOW) features (Vedaldi and Fulkerson, 2010), as done by Kiela et al. (2014). Dispersion scores generated from CNN features also seems like a worthwhile experiment.

The mmfeat (Kiela, 2016) library provides an excellent basis for starting our work on this. mmfeat has a pipeline for generating dispersion scores for two sets of images. We have adapted this pipeline to support images from our corpus, and to generate results for all images, the top 50, top 25, and top 10. We output our results to a JSON file so that we can easily aggregate them later. We also track the number of images for each word, so that we can ignore words that had 0 or 1 images.


```
{
        "top50": 0.39761144600075921,
        "top10": 0.38607501524692089,
        "all": 0.40102423120324254,
        "total_images": 94,
        "top25": 0.40653262346612978
}
```


Figure 42: Example dispersion results for a word

8.1 Proxy for concreteness

According to Kiela et al. (2014), these image dispersion scores for each word can be thought of as a proxy for concreteness. At a high level, we can reasonably assume that concrete words would have images that are very similar to each other, while abstract words would have images that are very different to each other. In order to validate that our results are reasonable, we need to compare them against concreteness scores generated for words via other means. Image dispersion scores range from 0 to 1, lower scores are more concrete and higher scores are more abstract.

For English, we can compare our image dispersion results against the two gold standard datasets we discussed in section 6: the University of South Florida norms (Nelson et al., 2004) and the University of Ghent dataset (Brysbaert et al., 2014). We use spearmans correlation to check if our results correlate with these two gold standard

datasets.

For our English superset (263,098 words), we compare our dispersion results against the gold standard datasets and found a weak (0.20-0.39), but consistent correlation. In table 6 we see the results: for all words, we scored -0.25, for the top 50 images, -0.24, for the top 25, around -.23, and for the top 10, around 0.19.

	USF dataset correlation	Ghent dataset correlation
All images	-0.242	-0.263
Top 50 images	-0.232	-0.258
Top 25 images	-0.223	-0.245
Top 10 images	-0.185	-0.199

Table 6: Spearman's correlation between our generated image dispersion scores and the gold standard concreteness scores

We also generated dispersion scores for our English words with Pyramid Histogram of Words (PHOW) features (Vedaldi and Fulkerson, 2010) using modified mmfeat code (Kiela, 2016) and an octave script but found no correlation between the scores and either dataset. We can't rule out there being a bug in our and/or their code when generating those PHOW features.

8.2 Pipeline integration

One approach place where we could integrate this with our current pipeline is to use the dispersion scores for foreign words to pick out the most concrete and abstract examples. We then try to take those results and re-compute our MRR and top N results for concrete and abstract words based on the threshold used by (Kiela et al., 2014), the median score.

Another idea would be to use the dispersion scores to help filter the words we include in our final results. We can find words whose normalized dispersion scores are some threshold away from where we expect them to be based on the gold standard concreteness ratings. We could then assume that any such cases would suggest a failure in the image search algorithm, which would invalidate those words from our dataset. Alternatively or additionally, we could try ignoring words whose image sets are all very different (they appear very abstract) since we dont have a great way of capturing multiple word senses. Our interest would be if our evaluation metrics improve after filtering

out this data.

8.3 Per language averages

We also calculated the average dispersion scores for each language in order to get a clearer picture of how similar the average image set is from one language to the next. In Table 7, we see that the per-language scores range from 0.459 to 0.515, which is relatively concise when we compare it to the per-language minimum and maximum dispersion scores. The overall average did not vary noticeably when using the top 10, 25, 50 or all images.

language	avg	min	max
Swahili	0.459	0.155	1.000
Chinese	0.465	0.122	0.758
Vietnamese	0.467	0.138	0.801
Somali	0.471	0.023	0.899
Hindi	0.474	0.180	0.714
Thai	0.475	0.229	0.760
Nepali	0.478	0.205	0.945
Gujarati	0.479	0.167	0.714
Uighur	0.482	0.234	0.910
Persian	0.482	0.219	0.670
Telugu	0.484	0.274	0.763
Azerbaijani	0.484	0.176	0.835
Tamil	0.486	0.263	0.942
Hungarian	0.486	0.187	0.836
Spanish	0.489	0.175	0.805
Bulgarian	0.490	0.136	0.781
Turkish	0.490	0.201	0.798
Uzbek	0.490	0.059	0.952
Serbian	0.494	0.148	0.809
Indonesian	0.494	0.172	0.828
Arabic	0.495	0.181	0.705
Cebuano	0.495	0.068	0.865
Bengali	0.495	0.244	0.970
Bosnian	0.496	0.213	0.764
Romanian	0.498	0.187	0.807
French	0.498	0.097	0.829
Albanian	0.498	0.232	0.809
Yoruba	0.501	0.003	0.998
Ukrainian	0.502	0.127	0.747
Filipino	0.502	0.220	0.827
Italian	0.502	0.112	0.848
English	0.503	0.002	0.996
Urdu	0.503	0.130	0.997
Dutch	0.507	0.129	0.824
Swedish	0.507	0.046	0.812
German	0.511	0.170	0.842
Slovak	0.512	0.191	0.835
Latvian	0.512	0.126	0.766
Welsh	0.515	0.165	0.856

Table 7: Average dispersion scores (using SIFT features) for all images. For context, we have also included the minimum and maximum dispersion scores.

9 Process

9.1 Computing resources

For most of our work, we use a distributed cluster (`nlpgrid`) run by the School of Engineering and Applied Sciences at the University of Pennsylvania. The system contains 12 nodes, each with 64 cores and 512GB of memory. It uses Open Grid Scheduler to manage the queue of jobs that users have requested the cluster to process. For some other work, such as generating our corpus and generating AlexNet CNN features, we use Amazon Web Services (AWS).

9.2 Preparation

Before we could start working with the images for any language, we had to extract the files from the package format we put them in upon creating the corpus. Before we could extract the files, we needed to transfer the package files to nlpgrid from Amazon `S3`. It takes several hours to download a 200 GB package file from `S3`, and then takes another 10 hours or so to extract a package file of that size.

Once the package file is fully extracted, we run a couple of processes on that package. The first process is to generate a package level report that provides useful statistics on the corpus. The package report runs completes in under an hour. A summary of some interesting fields for a few languages is in Table 8 while a full example is in Figure 58 in the Appendix.

	French	Spanish	Indonesian	Turkish	Arabic
Total size	250GB	200GB	199.4GB	174.4GB	130GB
Total images	962,222	959,099	946,444	984,243	941,011
Average file size	260KB	208.5KB	210.7KB	177.2KB	138.1KB
Average width	858	658	775	655	610
Average height	662	497	592	485	457
Median images per word	98	98	97	99	97
Number of unique hosts	246,610	196,571	252,514	150,937	60,254

Table 8: Summary statistics for packages in a few languages

We also generate a metadata-only version of the corpus, which takes significantly less space to store. It allows people to get a feel for how many images are in the corpus,

where they were sourced from, how large they are, etc. Finally, we generate a list of unique links of web pages where the images appear. This list of web pages is used to generate the complementary text corpus described in Section 5.

9.3 SIFT + HIST

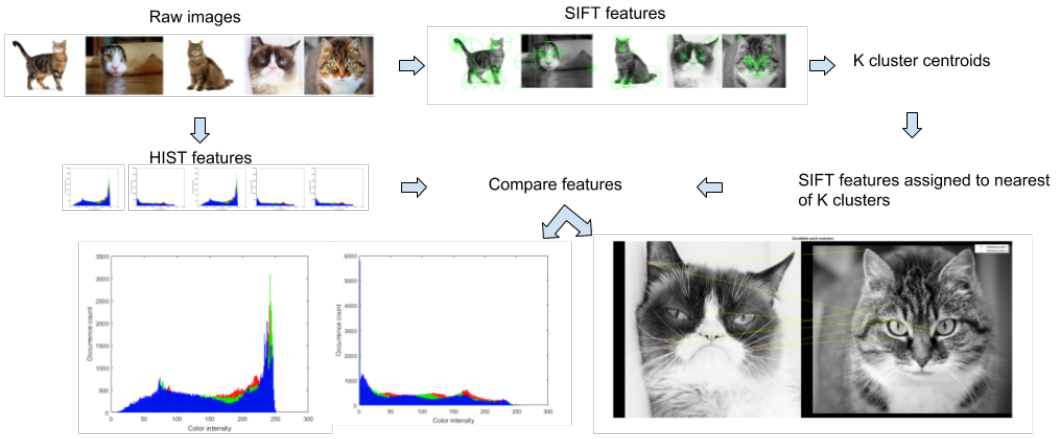

Figure 43: Big picture view of the SIFT+HIST pipeline.

The steps to generate MRR and top N accuracy results for SIFT and histogram features are shown in Figure 43 and are enumerated as follows:

1. The first step is to create raw SIFT keypoint features for all of the images in the dataset. We create one job per word that requests 4 cores and 32GB of memory, yielding 10,000 jobs in the typical case. The other step we take at the outset is the creation of histogram features, which are also split into one job per word with the same settings. Running the roughly 20,000 jobs for SIFT keypoint generation and histogram feature generation takes 4-6 hours. At this step, we output 2 new folders per word, one for SIFT features, one for HIST features. Inside these folders, we output 3 files per word, the raw SIFT and HIST features, and the normalized HIST features.

2. The next step we run for every language is the language-wide clustering of the SIFT keypoints. We run one job for this, requesting 1 full node in the cluster; 64

cores and 512GB of memory. With these settings, clustering takes approximately 1 day to run to completion. Our output at this step is a single file with 100 128-dimension cluster centroids.

3. After clustering completes, we need to run two batches of jobs that assign the each of the raw SIFT keypoints to the cluster closest to it. We create 1 job per word in the foreign language, as well as 1 job per possible translation word in english. This usually gives us another 20,000 jobs, and for each job we request 1 core and 8GB of memory. They take 2-4 hours to run to completion. The output at this step is one new folder for the English translations of the foreign words with files that contain the normalized cluster counts, in addition to normalized cluster counts for the image set of the foreign word.

4. Finally, with all of the features in hand, we run a single evaluation job to compute MRR and top N accuracy results, for which we request a full node in the cluster. The evaluation job takes approximately 36 hours to complete.

5. Our final output is a file that contains the `AVGMAX` similarity scores for all of the English words for each foreign word.

9.4 CNN

The steps to generate MRR and top N accuracy results for CNN features are shown in Figure 44 and are enumerated as follows:

1. The first step is to extract the `fc7` layer for all of the images in the dataset. Initially, we tried running one job per word with 16 cores and 128GB of memory, yielding 10,000 jobs in total. The 10,000 jobs took 3-4 days to run through the cluster. We then switched to running the 10,000 words on a single GPU-based system each. One system was local others running on `p2.xlarge` size `EC2` machines. In both GPU cases, the time to complete a language was 40 hours. We output one file per image at this step, which contains the 4,096 dimensional vector extracted from the `fc7` layer.

2. Because we were running the GPU-based jobs on systems external to our shared storage, we had to transfer our dataset into these servers, and the resulting features out of these servers and that added 10 hours to the process.

3. With all of the features in hand, we run a single evaluation job to compute MRR and top N accuracy results, for which we request a full node in the cluster. The evaluation job takes approximately 48 hours to complete.

4. Our final output is a file that contains the `AVGMAX` similarity scores for all of the English words for each foreign word.

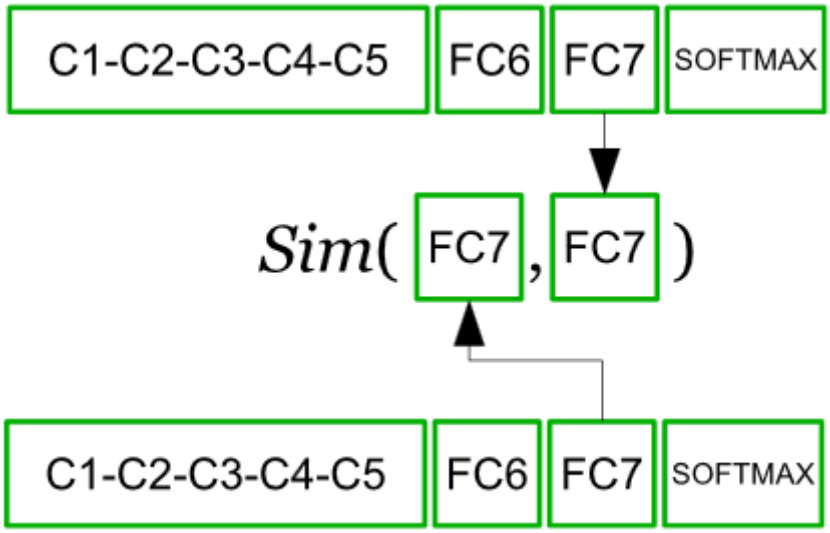

Figure 44: Big picture view of the CNN pipeline from (Kiela et al., 2015).

GPU systems (CUDA) and a highly-optimized implementation of 2D convolutions (cuDNN) massively improved performance at this step over distributed CPU-only implementations.

9.5 Image Dispersion

Generating image dispersion scores is a one step process. We adapted code from Kiela (2016) that generates SIFT features, runs k-means clustering and compares the images in a single process for each word. We create 10,000 jobs with 2 cores and 16GB of

memory. These 10,000 jobs take 3-5 hours to run through the cluster. We output a single `json` file for each word.

9.6 Scaling

Given the size of our corpus, at nearly every step of our process, we had to address concerns about the scale of our experiments. To run experiments against any foreign language, we had the pre-requisite of generating the features for the full set of potential English translations. We have 263,098 English words, yielding approximately 25 million images. Each foreign language adds 10,000 words and approximately 950,000 images. Therefore, we had to create SIFT, Histogram, and AlexNet features for 26 million images just to run our first full experiment for a foreign language. In total, for the 38 language subset, we created SIFT, Histogram, and AlexNet features for approximately 33 million images. To make this whole process tractable, we used several techniques to process our images. The first thing we did was to use Python's threading and multiprocessing libraries to process multiple images simultaneously. The next thing we did was to split our corpus into smaller chunks so that we could parcel the work out to the `nlpgrid` cluster. We broke the task down so that one word equaled one job, tuning the job parameters such that we maximized our throughput.

The next big area where we ran into scaling concerns was with loading the generated features into memory from disk for both the SIFT clustering and overall evaluation steps. Without any optimization, just loading the features into memory would take days, before we even start thinking about the time it would take to actually do something with those features. We found the best way to deal with this was to use Python's threading and multiprocessing libraries to massively parallelize loading the features into memory. We reserved a full 64 core node on `nlpgrid` for this task, and that along with the concurrency allowed us to load features for a language evaluation (20,000 words or 1.9 million features) in 5-8 hours. Later on in our experiments we did hit another scaling consideration that we had not thought of before. In an attempt to organize our feature files in a sane way, we stored them all on a single disk cluster. We started to see a

slowdown when 3 or more nodes on `nlpgrid` were loading features from the same disk cluster at the same time in a massively parallelized way. When this happened, simple `ls` commands that usually returned in less than a second took 15+ seconds.

The final area where we addressed scalability concerns was in computing the similarity between images using our features. To evaluate each foreign language, we compared approximately 1 million images for the foreign words against each of the 1 million images for English words (1 trillion similarities per language). The first optimization we made was heavily limit the number of times we called into `scikit-learn`'s `cosine_similarity` method. For each foreign word, we ran that method one time, comparing the 100 images against the full 1,000,000 images for all English words. The second optimization we made was to run those 100 x 1,000,000 cosine similarity calculations concurrently. Finally, we decided to use sparse matrices to limit our overall memory footprint. The SIFT and Histogram matrices were relatively sparse, and we were able to run 32 simultaneously without running out of memory. The CNN matrices were less sparse and we could only run 16 simultaneously without running out of memory. The nodes each have 512GB of memory.

10 Results

We replicated the experiments from previous research using features generated from our corpus and our results showed notable improvements. We also performed novel experiments on low-resource languages and after filtering to use language-confident images only.

10.1 Replicating previous work

We validated our dataset by replicating prior models for learning translations from images, and testing them on our dataset. We reproduced Bergsma and Van Durme (2011)'s SIFT and color histogram (SIFT+HIST) approach and Kiela et al. (2015)'s AlexNet-based (CNN) approach, testing them on the same five high resource languages that

were focus of prior work. For both approaches, we rank the English words as candidate translations based on their visual similarity with the foreign words (with different feature sets representing the images). We use the AVGMAX approach described in Section 7.4 to represent the similarity between each foreign word and English word. The number of candidate English words is equal to the number of entries in the bilingual dictionary after filtering. We evaluate the models' rankings using Mean Reciprocal Rank (MRR) and top-N accuracy.

dataset	**Bergsma**	**intersection**	**concrete**	**all**
# words	500	250	500	8,500
MRR	0.367	0.595	0.402	0.116
Top 1	0.311	0.53	0.341	0.09
Top 5	0.414	0.659	0.475	0.14
Top 20	0.537	0.764	0.577	0.186

Table 9: Our SIFT+HIST results compared to those reported by Bergsma and Van Durme (2011)

We compare 4 distinct scores using averages of 5 high resource languages (Dutch, French, German, Italian, Spanish).

1. Original results from prior research with their 500 word corpus
2. Our results using the 250 words that intersect our corpus and their corpus
3. Our results using the 500 most concrete words in our corpus for each language
4. Our results using the 8,500 non same translation words (stw)

Tables 9 and 10 show the results reported in previous work, along with our replication of their models on our dataset. Since our dictionaries contain different words than theirs, we intersect the words in our corpus and the words in their corpora, along with the 500 most concrete words in our dataset (since concrete nouns were the focus of previous work). The tables show the average score between Dutch, French, German, Italian, and Spanish. Contrary to the previous experiments, we have used only language-confident images and have filtered out same translation words in calculating our results.

dataset	**Kiela**	**intersection**	**concrete**	**all**
# words	500	250	500	8,500
MRR	0.658	0.829	0.653	0.277
Top 1	0.567	0.789	0.619	0.229
Top 5	0.692	0.872	0.71	0.326
Top 20	0.774	0.913	0.768	0.385

Table 10: Our CNN results compared to those reported by Kiela et al. (2015)

We successfully replicated the previous work, which shows that the concrete nouns in our corpus can serve as a reasonable basis for image-based similarity in BLI tasks. We highly outperformed previous results when using an intersection of words in the corpora (ours and theirs) in the results generated with SIFT and Histogram features in Table 9, as well as those generated with AlexNet features in Table 10. In looking at the 500 most concrete words in our corpus only, we achieve higher accuracies than they did on their 500 hand curated word set, which is better than we expected. We see a significant dip in our scores when we look at our full word sets, but this is fully in line with expectations given that we are also ranking translations for abstract words and there are many more distractors to choose from. We also confirm the previous finding that CNN features result in a substantial performance improvement over SIFT features.

10.2 Concrete word analysis

Unlike previous work, our corpus contains more than just concrete nouns. The third result column in Tables 9 and 10 compare the 500 most concrete words in our corpus versus for the full sets of words (which contain 8,500 words rather than 10,000 as a result of our filtering steps).

	ghent	usf	both	avg >.7	avg >.8
German	6661	3138	3079	831	553
Spanish	7002	2985	2926	890	605
French	6727	3141	3084	948	630
Italian	6413	3049	2994	915	605
Dutch	6189	2922	2868	922	654

Table 11: Breakdown for the number of English words from our corpus that appear in the concreteness corpora, with normalized scores (0-1) & specific thresholds

We used a relatively straightforward approach to find the 500 most concrete words

based on our two datasets with concreteness ratings (Nelson et al., 2004; Brysbaert et al., 2014). As a reminder, the USF dataset has concreteness ratings for 3,260 words, while the Ghent dataset has concreteness ratings for 39,954 words.

1. First we analyze the English words in our corpus, finding only those that appear in both of our datasets (USF & Ghent)

2. Next we normalize and average the concreteness scores of those words

3. Finally, we select the foreign words whose English translations have the highest concreteness score

Table 11 shows a breakdown of the number of our English translations that appear in the concreteness corpora, with normalized scores (0-1) & specific thresholds. In each of the five high resource languages, more than 500 words scored with this metric had concreteness ratings of 80% or more of the maximum score. We then picked the 500 foreign words whose English translations have the highest concreteness ratings in both datasets.

In order to better understand how the concreteness scores for this metric compared across languages, we calculated the average concreteness scores for each language we report results for in Table 12. The scores show relatively steady averages for the languages that are slightly above the middle possible score (3.5 for USF, 2.5 for Ghent, and 0.5 for the normalized average of the two). This means that the words in our datasets that we have coverage for all lean slightly concrete.

language	norm	USF	Ghent
Thai	0.67	4.79	3.64
Chinese	0.63	4.60	3.48
Vietnamese	0.63	4.47	3.45
Persian	0.62	4.52	3.30
Indonesian	0.61	4.45	3.23
Welsh	0.61	4.44	3.30
Azerbaijani	0.61	4.43	3.25
Uighur	0.61	4.41	2.95
Uzbek	0.61	4.38	3.31
Swahili	0.60	4.39	3.25
Hindi	0.60	4.40	3.18
Telugu	0.60	4.40	3.25
Tamil	0.60	4.39	3.22
Turkish	0.60	4.38	3.18
Latvian	0.60	4.36	3.19
Bengali	0.60	4.35	3.17
Bulgarian	0.60	4.36	3.18
Dutch	0.60	4.36	3.17
Urdu	0.59	4.33	3.17
Slovak	0.59	4.35	3.13
Italian	0.59	4.32	3.14
Filipino	0.59	4.32	3.15
Cebuano	0.59	4.30	3.16
Bosnian	0.59	4.32	3.17
French	0.59	4.32	3.10
Hungarian	0.59	4.30	3.09
Spanish	0.59	4.31	3.08
Nepali	0.59	4.29	3.17
Serbian	0.58	4.30	3.14
Albanian	0.58	4.28	3.09
Somali	0.58	4.27	3.23
Romanian	0.58	4.29	3.12
German	0.58	4.27	3.06
Swedish	0.58	4.26	3.11
Arabic	0.58	4.27	3.11
Gujarati	0.58	4.26	3.11
Ukrainian	0.57	4.23	3.07
Yoruba	0.56	4.14	3.10

Table 12: Average concreteness scores using USF, Ghent, and normalized average (norm) datasets.

10.3 Qualitative examples

In order to help understand what the results look like visually, we created image grids of the images for some foreign words in French and Indonesian. We included the images for the top 4 ranked English translation candidates along with the images for the foreign words.

French

Figure 45: Top 10 images for French word: *eau* and the top 4 ranked translations using SIFT+HIST features.

The first case we examine is the top 10 images for the French word *eau*, along with its top 4 ranked English translation candidates. Figure 45 shows a consistent theme where we see bodies of water, water droplets, and glasses of water. The set of images for *water* looks very similar to the French image set and having the correct translation as the top ranked option seems very reasonable. We think the near identical shapes are likely to have produced similar SIFT features, while the near identical colors led to a high degree of similarity in the color histogram features. Two other top options: *liquid* and *fluid* are both similar in meaning. While their image sets deviate slightly (like with the different colors for liquids), we still see similar shapes and colors on the whole. For the other top ranked option *ice*, it is also quite reasonable, given that it is another form of water. We do see some new shapes with the ice cubes but the colors are still nearly identical and therefore kept the overall similarity very high.

In the next example (Figure 46), we examine is the top 10 images for the French word *étoile* along with the top 4 ranked English translation candidates for it. For *étoile*,

Figure 46: Top 10 images for French word: *étoile* and the top 4 ranked translations using CNN features.

we see mostly drawings of stars along with some pictures of the night sky with stars visible. The top ranked translation *star* has the exact same theme, and we suspect that the star shapes and star-like dots in the night sky led to the similarity in the SIFT features. The prevalence of yellow colored star shapes also seems to have contributed to similarity on the color histogram front. We see the roles reversed in the second option for *stars*, having mostly pictures of the night sky and just a few star shapes. We expect the dots in the night sky provided a lot of shape-based similarity for the SIFT features, while the colors in the night sky also were very similar. For the next option of *point*, we think the dots were similar shape-wise to the starts in the night sky, and therefore contributed to the SIFT similarity. We see an interesting scenario for *points* where we are getting lots of image results for the sense of the word that reflects shopping rewards points. These images have lots of images with star shapes in them, which surely boosted the similarity of the SIFT-based features.

Indonesian

The first example (Figure 47) we look at for Indonesian is one where the SIFT+HIST features did a reasonably good job of finding candidate translations. For the Indonesian word *cahaya*, that translates to *light*, we do get the correct translation in the lead. The image set for *cahaya* shows a relatively consistent picture of bright lights, many from

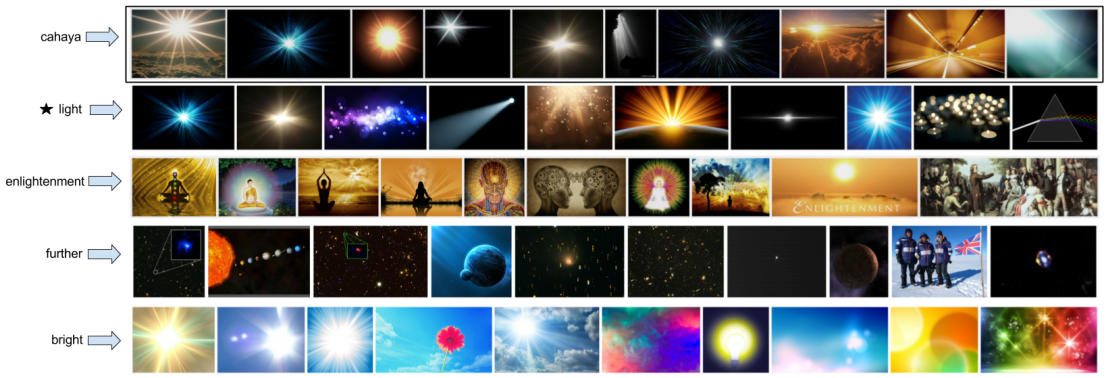

Figure 47: Top 10 images for Indonesian word: *cahaya* (light) and the top 4 ranked translations using SIFT+HIST features.

stars including our sun. When we look at the image set for *light*, we also see many stars and other pictures of bright lights. We expect that the shapes of the stars produced similar SIFT features, and the background of the dark sky with a bright star in the middle produced many similar colors in the histogram features. One note about these images is that we do see what appear to be identical images in the sets for English and Indonesian. While they do appear to be derived from the same source image, we did confirm that the images were present on websites in the correct language, and were not exact matches (different scale/quality/etc). In the image sets for two of the other words (*further* and *bright*), we have some pictures of the (day) sky with bright lights, as well as pictures of the night sky with a star showing. We expect that these still produced a strong degree of shape and color similarity for the SIFT and color histogram features respectively. On the surface, the images for the other top ranked option (*enlightenment*) look much different, however when we think about what the underlying comparisons are of, we can see why they would rank highly for similarity. For the SIFT features, we are also focused on bright circular objects with light rays protruding outwards. We also see many bright colors from lights that are shining on the human-like shapes in the images, which likely boosted the color histogram similarity.

We also look at the SIFT+HIST results (Figure 48) for the Indonesian word *kucing*, which translates to *cat*. In the image set for *kucing*, we see a relatively consistent series of images that contain one or more cats in them. Looking at the top ranked option, *golden*, the image set shows exclusively golden retriever dogs. Looking at the colors

Figure 48: The top 4 ranked translations of the Indonesian word *kucing* using SIFT+HIST features. Compare to the rankings produced with our CNN features in Figure 49.

of the dogs (which look similar to the cats in *kucing*) and the grassy backgrounds, we think that there was a high degree of similarity color-wise. We also think the shapes of the cats and the golden retrievers are similar at a high level and expect a decent degree of similarity in the SIFT features as well. Looking at the next option, *shellfish*, the shapes don't look very similar to us, and we expect similar colors in the pictures of shellfish contributed to the similarity for this word. In the case of *mammals*, it seems like a reasonable option on the surface, and when we look at the shapes (some of big cats), we can see why there would be some similarities in the SIFT features. We also think the colors in the grassy backgrounds were similar and added similarity in the histogram features. The correct translation, *cat*, is ranked fourth, and we think the shapes in particular appear very similar, however the colors the cats themselves and the backgrounds (not grassy) are notably different.

In Figure 49, we examine the same word (*kucing*) with CNN features. We see immediate benefits from the CNN features, with the correct translation *cat* jumping to the top. We expect the AlexNet features that activate strongly for cats led to a high degree of similarity here. We also notice the image sets for two of the other top ranked words (*persian* and *pet*) contain many pictures of pet cats. In the case of *persian*, we are honing in on a specific cat breed, while in the case of *pet*, it is showing cats and dogs together. We expect in both of these cases, features that relate to cats were activating very strongly. Finally, *animal* also seems like a reasonable option to get as a top ranked

Figure 49: Our dataset allows translations to be discovered by comparing the images associated with foreign and English words. Shown here are five images for the Indonesian word *kucing*, along with its top 4 ranked translations using CNN features. Compare to the rankings produced with our SIFT+HIST features in Figure 48.

translation candidate. We see some pictures of big cats, and other animals that have some features that might be cat-like.

Abstract examples

Concrete words yield fairly easy to understand results visually. We were curious to see if visual analysis of abstract words would provide similar insights. We examine the CNN-based results for two Indonesian words to try and better understand what may be happening with abstract words.

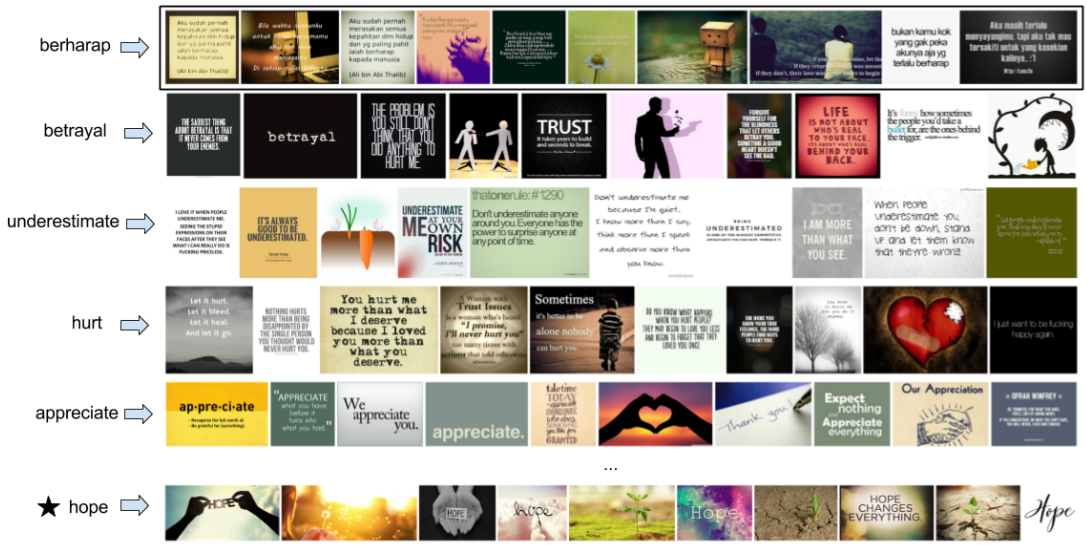

Figure 50: Shown here are the top 10 images for the abstract Indonesian word *berharap*, along with its top 4 ranked translations using CNN features. At the bottom are images for the actual translation *hope*, which was ranked 536.

In the first example (Figure 50), we look at the Indonesian word *berharap*, whose English translation is *hope*. When examining the images for *berharap*, we see examples of various sayings written in Indonesian about hope. The backgrounds, fonts, and text colors vary a lot for he writings. When we look at the top 4 ranked English translation candidates (*betrayal*, *underestimate*, *hurt*, *appreciate*), the words don't have much similarity meaning-wise, and the images all contain sayings in English about each respective word. We expect that there are some CNN features that activate very strongly when there is text present, though it is also probable that some of the text backgrounds have features are slightly more similar for these words than others. Looking at the image set for the actual translation *hope*, we see English text with the word itself (not sayings about it) as well as some bright lights, new growths, and other things that may symbolize hope. Given how different the image sets are, we are not surprised that the real translation ranked above 500. It also shows some of the difficulties in using images for abstract words to represent similarity, when the top images for the Indonesian searches mostly include longer sayings and the top images for the English searches show both the word itself and much more symbolism.

Figure 51: Shown here are 10 images for the abstract Indonesian word *konsep*, along with its top 4 ranked translations using CNN features. The actual translation, *concept*, was ranked 3,465.

In the next abstract word example (Figure 51), we see a completely different problem as to why the correct translation ranks very low. We examine the Indonesian word

konsep, which translates to *concept*. When we look at the top images for *konsep*, we see various attempts to make visually concrete what a concept might look like. Looking down at some of the top ranked results, we see some similarities in some of the images. For example, when we look at *oriented*, we see a very similar looking dartboard and some diagrams that appear similar visually to the bubbles around the woman's head. In the image set for *department*, we see silhouettes of humans, which probably activated AlexNet features for humans, and might have been similar to the *konsep* features that were activated for the human head. In the image set for *gifted*, we see drawings and pictures of children, where the same human and human-head based features were activating strongly on both sides. We see a different set of images that look similar for *top level*. The three apples from *konsep* looks similar to the stacked metal balls. We expect they both strongly activated the features for round objects, which increased the similarity. Looking at the image set for the actual translation (ranked over 3,000), we immediately notice why it did yield much similarity: all of the pictures are of concept cars. In this image set, the sense of the word is actually concrete , as the abbreviated version of *concept car*, whereas Indonesian does not use *konsep* as an abbreviation for *mobil konsep*, the direct translation. When we tested searching for *mobil konsep*, the image set looked much more similar and we expect the English word *concept* would have been a top rank for it.

10.4 Results for 38 language subset

Table 13 presents the performance results for a wide range of languages, varying in the amount of resources that are typically available for MT systems. We find good performance in high resource languages like French, Spanish, Arabic, as well as in some lower resource languages like Indonesian and Turkish. We also find much poorer results for other very resource languages like Gujarati and Somali.

An important note is that we noticed in the translations for our dataset and for the previous dataset was that there was a significant number of words whose translation was an exact match. When we used the raw translations without filtering, several lan-

guages exceeded expectations by a large margin. We inspected the bilingual dictionaries to figure out how many same translation words were there for each language, and we found a correlation between the languages that scored higher than we expected and the languages that had a relatively high number of the same translation words (stw). To counter-act this, we took an additional filtering step of generating results with those words omitted. We also included the set of possible English translations in each language's bilingual dictionary, to provide context when looking at the scores.

method	All words		500 most concrete		**eng words**
	S+H	**CNN**	**S+H**	**CNN**	
Spanish	0.182	0.382	0.495	0.724	9,918
Persian	0.156	0.323	0.212	0.374	950
French	0.17	0.321	0.385	0.61	10,483
Thai	0.14	0.32	0.229	0.46	5,555
Chinese	0.131	0.317	0.222	0.397	4,352
Dutch	0.131	0.311	0.353	0.641	10,528
Italian	0.109	0.262	0.382	0.651	8,926
German	0.113	0.254	0.396	0.636	10,064
Bulgarian	0.115	0.253	0.327	0.565	8,605
Indonesian	0.104	0.244	0.35	0.618	10,297
Turkish	0.113	0.224	0.373	0.567	10,231
Serbian	0.118	0.218	0.216	0.488	8,264
Swedish	0.097	0.207	0.287	0.517	9,565
Arabic	0.088	0.207	0.265	0.51	10,189
Vietnamese	0.1	0.196	0.233	0.46	6,633
Romanian	0.088	0.193	0.245	0.467	9,118
Hungarian	0.076	0.189	0.278	0.53	10,945
Bosnian	0.084	0.184	0.265	0.47	7,494
Ukrainian	0.078	0.17	0.277	0.456	4,964
Uighur	0.138	0.165	0.028	0.112	21,250
Slovak	0.064	0.159	0.217	0.446	6,539
Latvian	0.057	0.153	0.152	0.447	7,102
Hindi	0.062	0.147	0.15	0.442	9,446
Cebuano	0.082	0.131	0.096	0.263	7,745
Albanian	0.053	0.124	0.157	0.333	6,043
Uzbek	0.085	0.119	0.058	0.106	12,416
Azerbaijani	0.053	0.118	0.175	0.425	6,281
Filipino	0.045	0.106	0.125	0.401	9,772
Urdu	0.039	0.091	0.105	0.283	11,128
Bengali	0.028	0.088	0.128	0.328	12,487
Nepali	0.038	0.086	0.087	0.254	11,633
Tamil	0.031	0.077	0.1	0.289	9,888
Swahili	0.028	0.056	0.077	0.178	7,167
Telugu	0.025	0.051	0.042	0.17	9,577
Yoruba	0.036	0.049	0.015	0.029	1,585
Welsh	0.027	0.048	0.043	0.149	7,567
Gujarati	0.013	0.041	0.041	0.181	12,026
Somali	0.03	0.039	0.031	0.082	8,617

Table 13: MRR scores for additional languages filtering out same translation words (stw). The results are sorted so the best MRR score with CNN features for all words is at the top. For comparison purposes, we included the number of possible English translations for each language.

Another interesting way to slice the results is to look at the languages that had the largest % gain when between the SIFT+HIST results and the CNN results (Table 14).

We see the strongest gains when looking at lower resource languages like Gujarati and Bengali, however the improved MRR scores for those languages still don't appear to have a lot of signal. Aside from a few lower resource outliers, we see across the board improvement of 100% or more the languages in our set. If there is signal to be found, the CNN based approach does a consistently better job of finding it than the SIFT+HIST based approach.

method	All words			500 most concrete		
	% gain	**S+H**	**CNN**	**% gain**	**S+H**	**CNN**
Gujarati	215.38	0.013	0.041	341.46	0.041	0.181
Bengali	214.29	0.028	0.088	156.25	0.128	0.328
Latvian	168.42	0.057	0.153	194.08	0.152	0.447
Hungarian	148.68	0.076	0.189	90.65	0.278	0.53
Slovak	148.44	0.064	0.159	105.53	0.217	0.446
Tamil	148.39	0.031	0.077	189	0.1	0.289
Chinese	141.98	0.131	0.317	78.83	0.222	0.397
Italian	140.37	0.109	0.262	15.97	0.382	0.651
Dutch	137.4	0.131	0.311	48.16	0.353	0.641
Hindi	137.1	0.062	0.147	194.67	0.15	0.442
Filipino	135.56	0.045	0.106	220.8	0.125	0.401
Swedish	135.23	0.088	0.207	59.25	0.265	0.51
Indonesian	134.62	0.104	0.244	18	0.35	0.618
Albanian	133.96	0.053	0.124	112.1	0.157	0.333
Urdu	133.33	0.039	0.091	169.52	0.105	0.283
Thai	128.57	0.14	0.32	100.87	0.229	0.46
Nepali	126.32	0.038	0.086	191.95	0.087	0.254
German	124.78	0.113	0.254	30.56	0.396	0.636
Azerbaijani	122.64	0.053	0.118	142.86	0.175	0.425
Bulgarian	120	0.115	0.253	72.78	0.327	0.565
Romanian	119.32	0.088	0.193	90.61	0.245	0.467
Bosnian	119.05	0.084	0.184	77.36	0.265	0.47
Ukrainian	117.95	0.078	0.17	64.62	0.277	0.456
Arabic	113.4	0.097	0.207	80.14	0.287	0.517
Spanish	109.89	0.182	0.382	12.53	0.495	0.724
Persian	107.05	0.156	0.323	76.42	0.212	0.374
Telugu	104	0.025	0.051	304.76	0.042	0.17
Swahili	100	0.028	0.056	131.17	0.077	0.178
Turkish	98.23	0.113	0.224	11.8	0.373	0.567
Vietnamese	96	0.1	0.196	97.42	0.233	0.46
French	88.82	0.17	0.321	35.58	0.385	0.61
Serbian	84.75	0.118	0.218	125.93	0.216	0.488
Welsh	77.78	0.027	0.048	246.51	0.043	0.149
Cebuano	59.76	0.082	0.131	173.96	0.096	0.263
Uzbek	40	0.085	0.119	82.76	0.058	0.106
Yoruba	36.11	0.036	0.049	93.33	0.015	0.029
Somali	30	0.03	0.039	164.52	0.031	0.082
Uighur	19.57	0.138	0.165	300	0.028	0.112

Table 14: MRR scores for additional languages filtering out same translation words (stw). The results are sorted by the largest percentage gain for all words.

10.5 Language-confident results on 11 language subset

Table 15 presents the performance results for a subset of the languages in Section 10.4, which also vary in the amount of resources that are typically available for MT systems. We find good performance in high resource languages like Spanish, French, Dutch, as well as in some lower resource languages like Indonesian and Turkish. We also find much poorer results for other very resource languages like Bengali and Uzbek. As with before, we have filtered out same translation words from our result sets.

method	non-stw words		500 most concrete	
	S+H	**CNN**	**S+H (c)**	**CNN (c)**
Spanish	0.127	0.322	0.436	0.711
French	0.136	0.288	0.396	0.61
Dutch	0.11	0.286	0.38	0.663
Italian	0.105	0.25	0.405	0.676
Indonesian	0.1	0.238	0.353	0.634
German	0.103	0.237	0.392	0.637
Turkish	0.086	0.196	0.321	0.554
Arabic	0.068	0.162	0.224	0.466
Cebuano	0.022	0.129	0.049	0.254
Bengali	0.027	0.081	0.123	0.304
Uzbek	0.049	0.068	0.036	0.085

Table 15: MRR scores for language-confident images only with AlexNet (CNN) and SIFT+HIST (S+H) features

As in Section 10.4, we show the results sorted by the largest % gain between SIFT+HIST results and CNN results (Table 16. Again, we see the strongest gains when looking at lower resource languages like Cebuano and Bengali. The improved MRR scores for those languages still don't appear to have a lot of signal. Aside from a single outlier (Uzbek), we see across the board improvement of 100% or more the languages in our set. Using only language-confident images, the CNN features still does a much better job of helping us find translations than SIFT+HIST based features.

method	All words			500 most concrete		
	% gain	S+H	CNN	% gain	S+H	CNN
Cebuano	486.36	0.022	0.129	418.37	0.049	0.254
Bengali	200	0.027	0.081	147.15	0.123	0.304
Dutch	160	0.11	0.286	74.47	0.38	0.663
Spanish	153.54	0.127	0.322	63.07	0.436	0.711
Arabic	138.24	0.068	0.162	108.04	0.224	0.466
Italian	138.1	0.105	0.25	66.91	0.405	0.676
Indonesian	138	0.1	0.238	79.6	0.353	0.634
German	130.1	0.103	0.237	62.5	0.392	0.637
Turkish	127.91	0.086	0.196	72.59	0.321	0.554
French	111.76	0.136	0.288	54.04	0.396	0.61
Uzbek	38.78	0.049	0.068	136.11	0.036	0.085

Table 16: MRR scores for additional languages filtering out same translation words (stw). The results are sorted by the largest percentage gain for all words.

10.6 Precision at N

While we focused overall on MRR scores, as they provide an all inclusive view of how translations are ranked in our results, we still think it is valuable to look at the results for the top N results at different thresholds.

method	All words			500 most concrete		
	Top 1	Top 5	Top 20	Top 1	Top 5	Top 20
Spanish	0.264	0.383	0.45	0.675	0.766	0.823
Dutch	0.237	0.336	0.395	0.612	0.725	0.777
French	0.24	0.339	0.393	0.578	0.665	0.719
Italian	0.205	0.297	0.354	0.638	0.743	0.804
German	0.197	0.277	0.334	0.599	0.716	0.774
Indonesian	0.197	0.28	0.333	0.585	0.706	0.797
Turkish	0.156	0.236	0.287	0.513	0.626	0.7
Arabic	0.126	0.198	0.246	0.428	0.536	0.644
Cebuano	0.112	0.143	0.165	0.224	0.293	0.375
Bengali	0.058	0.102	0.138	0.255	0.383	0.495
Uzbek	0.057	0.075	0.097	0.05	0.12	0.191

Table 17: Top 1,5,20 scores for additional languages filtering out same translation words (stw), using CNN features. The results are sorted by the highest Top 20 score for all words.

In Table 17, we show the language-confident precision at N scores for the correct translation appearing in the top 1, 5, and 20 results for CNN features. We show for the case of all words (minus same translation words) and the 500 most concrete words for each language.

10.7 CNN improvement examples

French

In Table 18, we show a selection of the French words where we had the largest gains in the rank of the correct translation between SIFT+HIST features and CNN features.

French word	English word	Change in rank	SIFT+HIST rank	CNN rank
comtesse	countess	+5001	5002	1
gnral	general	+4962	4963	1
mythe	myth	+4932	4953	21
martienne	martian	+4882	4956	74
royaume	kingdom	+4828	4855	27
romain	roman	+4635	4636	1
niveau	level	+3722	3723	1

Table 18: Examples of some French words where we saw large gains in rank between SIFT+HIST and CNN results.

For one of the examples, we look a little deeper. Specifically, we look at the top 10 images for the French word vs. the top 10 images for the 4 English words that were most highly ranked by similarity. We chose the French word *romain*, which translates to *roman*.

Figure 52: The top 4 ranked translations of the French word *romain* using SIFT+HIST features. Compare to the rankings produced with our CNN features in Figure 53. This is a case where CNN scores massively improved on the SIFT+HIST scores.

The SIFT+HIST feature version is shown in 52. The image set for *romain* paints a fairly consistent picture, with pictures or drawings of roman soldiers in red uniforms

with metal armor. Looking at the images for two of the words (*carried* and *frequented*), it doesn't make a whole lot of sense to us why they rank highly. However, when looking at the image set for *knight templar*, we think the men in armor have similar shapes to the roman soldiers in their own armor and therefore generated similar SIFT features. We also think that the red uniform on the roman soldiers and the red cross for the knights could show highly in the histogram features. In the case of the image set for the word *monk*, we suspect the similarity is down to shapes common in pictures of humans and the reddish colors in the clothing of the monks.

Figure 53: The top 4 ranked translations of the French word *romain* using CNN features. Compare to the rankings produced with SIFT+HIST features in Figure 52. This is a case where CNN scores massively improved on the SIFT+HIST scores.

We show the version based on CNN features in shown in 53. The image set for *romain* stays the same, with pictures or drawings of roman soldiers in red uniforms with metal armor. In this case, the top ranked result comes back as *roman*, and in the image set for the English word, we do see more drawings of roman soldiers, as well as drawings of roman citizens, pictures of the coliseum and a map of the Roman empire. It is interesting to see a slightly less soldier-centric image set for English than for French, and we wonder if it has something to do with France's territory once belonging to the Roman empire. While the other top translation options don't seem too similar to the correct translation, if we look for visual similarities between the images and think about what features AlexNet might be picking up on, we can get some idea. For *king*,

it seems like the helmets of the soldiers and the king's crown could activate strongly for the same features. Likewise, the scepter in the king's hand could fire for the same features as the sword in the soldier's hand. We suspect that in the cases of *human* and *merchant*, features for the human shapes and faces played a prominent role in the similarity. Overall, the deeper CNN-based features helped significantly in finding the correct translation, probably due to features that strongly activate in humans, weapons, and armor.

Indonesian

In Table 19, we show a selection of the Indonesian words where we had the largest gains in the rank of the correct translation between SIFT+HIST features and CNN features.

Indonesian word	English word	Change in rank	SIFT+HIST rank	CNN rank
silinder	cylinder	+5103	5104	1
laksamana	admiral	+5079	5081	2
fisika	physics	+4983	5067	84
jendral	general	+4905	4907	2
naga	dragon	+4786	4787	1
komik	comic	+4733	4740	7
merpati	pigeon	+3815	3816	1

Table 19: Examples of some Indonesian words where we saw large gains in rank between SIFT+HIST and CNN results.

For one of the examples, we look a little deeper. Specifically, we look at the top 10 images for the French word vs. the top 10 images for the 4 English words that were most highly ranked by similarity. We chose the Indonesian word *naga*, which translates to *dragon*.

The SIFT+HIST feature version is shown in 54. The image set for *naga* paints a fairly consistent picture, with drawings of dragons in various backgrounds. Two of the words that appear in the top ranks, *napoleon* and *abraham* don't make a whole lot of sense to us. The only similarities we can glean is that some of the images appear to be drawn on similar backgrounds. In the case of a third word (*shaman*), the backgrounds of the images look even more mythical, and we suspect again that the color similarities played a strong role. The other top ranked word's image set (*creatures*) actually seems

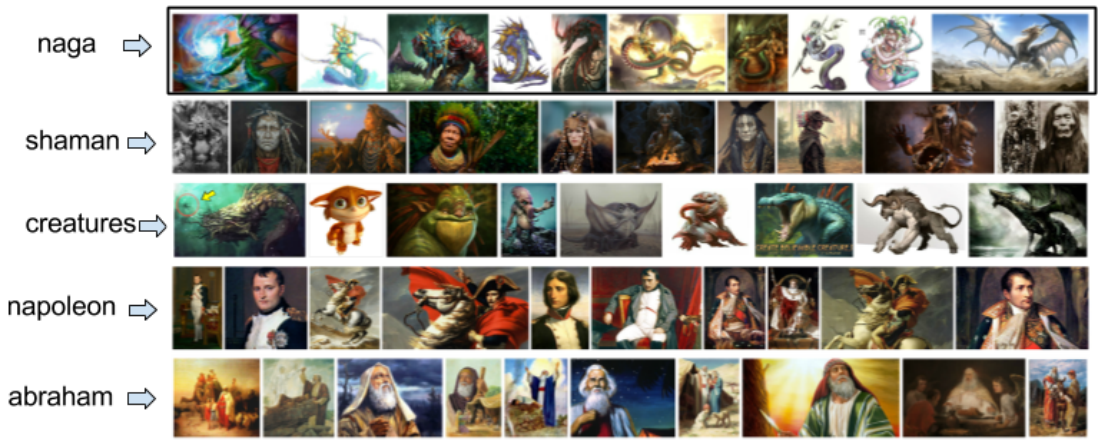

Figure 54: The top 4 ranked translations of the Indonesian word *naga* using SIFT+HIST features. Compare to the rankings produced with our CNN features in Figure 55. This is a case where CNN scores massively improved on the SIFT+HIST scores.

somewhat reasonable, both from the images we see and the meaning of the word. We see mythical creatures, some of which are dragons, others of which are lizard-like. We expect that the shapes and colors both contributed to the similarity for *creatures*.

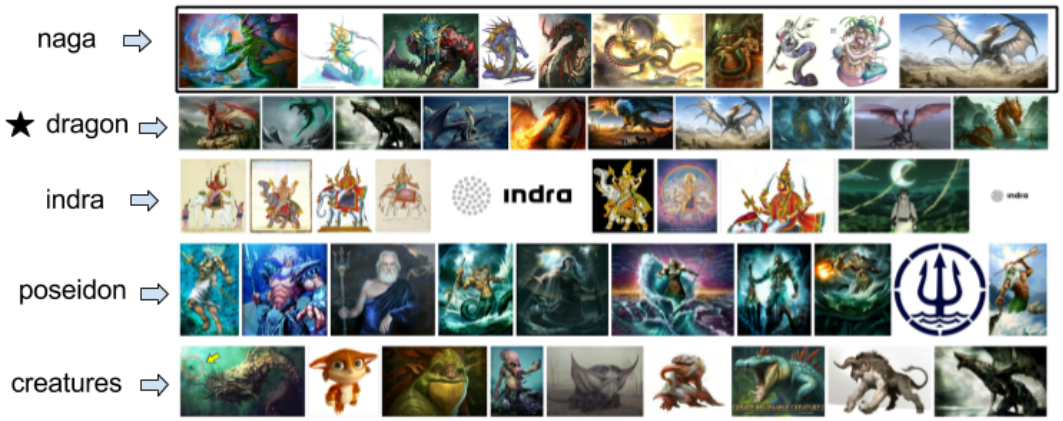

Figure 55: The top 4 ranked translations of the Indonesian word *naga* using CNN features. Compare to the rankings produced with SIFT+HIST features in Figure 54. This is a case where CNN scores massively improved on the SIFT+HIST scores.

We show the version based on CNN features in shown in 55. The image set for *naga* stays the same, with drawings of dragons in various backgrounds. In this case, the top ranked result comes back as *dragon*, and in the image set for the English word, we also see a string of pictures of dragons. One interesting difference comes to light when we compare them, the dragon images for *naga* seem to be slightly more water-based than

the dragon images for *dragon*. We wonder if this has to do with Indonesia being an island nation and their mythical dragons taking a bias to the local environment. This also might explain why one of the top 4 results we see is the word *poseidon*, where the image set shows primarily pictures of the Greek god, oftentimes with a serpent-like bottom, which is likely to trigger similar features to the more water-bound Indonesian dragons that also have serpent-like bottoms. We also see *creatures* again in the top results, likely because both sides had features that activate strongly for lizards. Finally, when we look at the results for *indra*, we suspect that the similarity might be coming about due to the elephant's trunk being serpent-like and/or the many arms of the Hindu god being similar looking to a dragon's claws or wings. The deeper features, and most likely those relating to lizards and serpents, helped significantly in making the correct translation the most similar option.

10.8 CNN decrease examples

While on average, the ranks of CNN results were lower than SIFT+HIST results, we also found some cases where the rank for the CNN results went massively downwards in comparison. We show a sampling of these words in Table 21. We do note that most of these decreases are not by words that appeared in the top results overall. Most of the examples we found were no higher than rank 150 with SIFT+HIST features. Nonetheless, we think it is informative to examine these results in closer detail so we can understand why this might have happened.

Table 20: French decrease examples

French word	English word	Change in rank	SIFT+HIST rank	CNN rank
pens	thought	-4646	500	5146
jumelles	twins	-4204	658	4862
femelles	females	-4074	169	4243
dispersion	dispersal	-3659	1486	5145
luimłme	himself	-3604	179	3783
fantastique	fantastic	-3549	1549	5143
noblesse	nobility	-2423	177	2600

Table 21: Examples of some French words where we saw a large decrease in rank between SIFT+HIST and CNN results. These were fewer and farther between, as the overall scores improved heavily, but are still worth noting.

The SIFT+HIST feature version is shown in 56. The image set for *noblesse* paints a fairly consistent picture, though not one we would have expected. Instead of pictures of things that might relate to nobility, we see pictures from an anime series named noblesse. As such, some of the results we see actually make quite a bit of sense. In the image set for the word *mangas*, we find pictures from other anime that have similar profiles of characters with similar colors. To a lesser extent in the image set for *the plot*, we find the words on top of images that are also from anime. The image sets for the other two words, *adversary* and *vengeance* both have some drawings in styles other than anime of human characters, other than that though, we don't see a lot of similarity.

Figure 56: The top 4 ranked translations of the French word *noblesse* using SIFT+HIST features. Compare to the rankings produced with our CNN features in Figure 57. This was a more unusual case where the SIFT+HIST features outperformed the CNN features.

In 57, we show the version based on CNN features. The image set for *noblesse* stays the same. The top result we get back using CNN features is very apt: *anime*. There are lots of drawings of faces, and we suspect there are some features in AlexNet that trigger very strongly for scenes from anime. We see *mangas* again as well, which seems likely to activate strongly for those same features. We also see *sasuke*, a popular character from an anime show, and *bleach*, a popular anime show, which are likely showing a high degree of similarity for the same reasons as *anime* and *mangas*. With *bleach* specifically, it is interesting to see that another meaning of the word, bleach as in the chemical, is mostly washed out from the image set by the results from a popular tv series by the same name. Finally, when we look at the image set for *nobility*, we see

pictures for individual nobles or groups of nobles, which is what we would expect to see. It appears that *noblesse* is the name of the anime series for both English and French speaking viewers, they have just co-opted a French word, which explains why we don't see results for that show in the image set for *nobility*. This turns out to be another case where grouping the images for each sense of the word together would be useful.

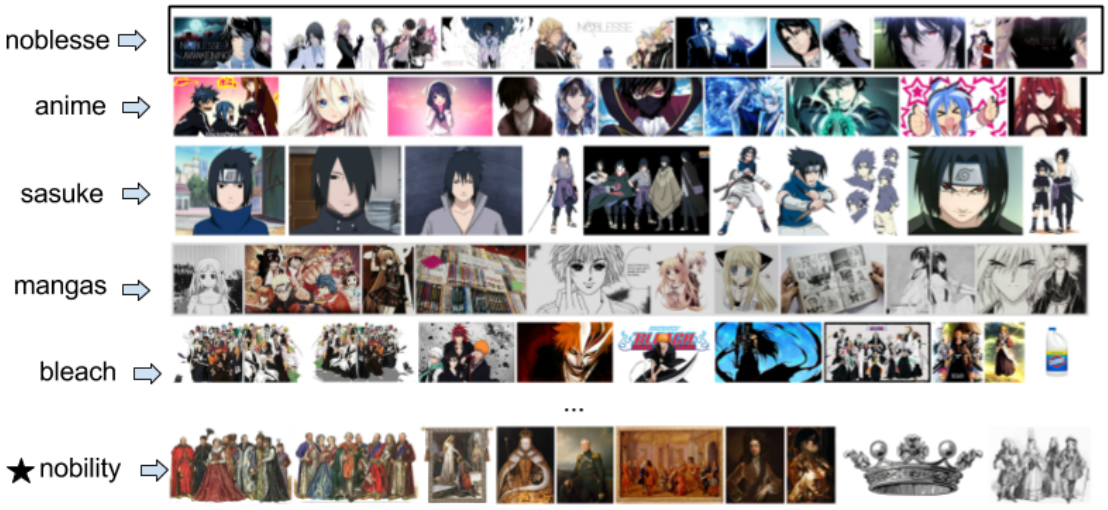

Figure 57: The top 4 ranked translations of the French word *noblesse* using CNN features. Compare to the rankings produced with SIFT+HIST features in Figure 56. This was a more unusual case where the SIFT+HIST features outperformed the CNN features.

11 Conclusion

Our results compare favorably with past research. We see an across the board improvement for the 5 high resource languages previously experimented on (Bergsma and Van Durme, 2011; Kiela et al., 2015). We see these improvements even when filtering out same translation words and only using language-confident images (something past research did not do). We think part of the improvement is down to improved quality of Google image search results, the previous corpora was created in 2011, 5 years earlier than ours.

We see a mixed picture in our results for other languages (excluding same translation words). Some like Indonesian, Arabic, Turkish perform comparably to the 5 high

resource languages (Dutch, French, German, Italian, Spanish). Others like Somali, Gujarati, and Telugu performed quite poorly in comparison. We also see a minor drop in the MRR scores when we re-evaluate our results using language-confident images only.

For the full set of words vs. the most concrete words in the group, we also see an across the board dip in the scores. However, this is in line with our expectations, as abstract words appear more difficult to compare from qualitative analyses. Another reason the results decline for the full set of words is that there are more potential distractors in the mix. Regardless, we still think that there is adequate signal in our dataset to enhance the Bilingual Lexicon Induction task.

In the course of our research, we have also created a large-scale image resource for bilingual lexicon induction for words of arbitrary parts of speech in 100 languages. The corpus is substantially larger than any before it, in terms of languages, image count, word count, and word variability. In replicating prior work using SIFT, Histogram and CNN features, we have demonstrated the utility of the corpus. We also feel that the number of languages and types of language (many low-resource languages) provide a basis for highly varied experiments. Along with the images we provide a complementary text corpus of substantial size, extracted from the web pages from which the images were drawn. Using only images from webpages in the expected languages provides another useful resource for BLI tasks.

Our dataset (25TB raw) will be distributed via the LDC, with memory compact featurized versions available for download. Links to data samples will be made available for free. Our code has also been open sourced [10], allowing the other researchers to easily create their own corpora for other source dictionaries. We will also open source the code used to run our experiments.

12 Future work

There are a multitude of ways our work could be extended and improved upon. Some of the more obvious ways are to use new and different techniques that might be better

[10] https://github.com/brendandc/multilingual-google-image-scraper

than the ones we used. For example, we could use the newer and better CNN models like ResNet (He et al., 2016) to generate image features. We could also see how the results compare when using different numbers of images per word. We could also try additional comparison metrics between words and translations. We could try using Jaccard or Dice similarity instead of cosine similarity, or try swapping `AVGMAX` for MAXMAX or AVGAVG.

We might also want to improve the quality of the source data, by filtering and re-categorizing. We could try disambiguating various senses of words with images (i.e. ImageNet), creating a distinct set of images for each sense. We could use human annotators to assist us to that end, or also use them to help us filter out images that don't belong in the set. We could try clustering similar images for a word together and have human annotators assign the word senses. Yet another thing we could try is to identify the relevant portions of images (via bounding boxes) so those portions alone can be used for the comparison. We could also try improving our language-detection results with more advanced language detection utilities.

Another thing we could try is to run new experiments with each of our foreign languages, comparing against all 263,098 English words in our English superset. Using a much larger number of distractors (possible translations) would be a more realistic experiment, given the set of potential translation words that exist in the real world. We would have to use top-N accuracy, and our expectation is that the scores would tank. Further, it would be interesting to see how the advantage that concrete words hold over the larger set holds with more distractors in the mix.

The last and perhaps most interesting extension of our work is to try combining image-based features with state of the art text-based features for the bilingual lexicon induction task. One idea is that we could use text-only or image-only features to provide an initial ranked list of translations candidates and then the other option could re-rank them. Another idea is that we could find and extract the caption text directly associated with the images and use it to create text features, or use image features to support the translation task for image captions, as (Hitschler et al., 2016) did for Ger-

man and English only. Future work could also exploit the image-text parallel properties of our dataset (Vulić et al., 2016). Furthermore, image dispersion scores can be used as weights for the image similarity calculation, where less disperse image sets would get a higher weight than more disperse image sets. Using common or similar images as a bridge, associations could be drawn between the text of web pages across 100 languages. These associations might be useful for distributional methods in bilingual lexicon induction and multilingual semantics tasks.

A Appendices

A.1 Size of the bilingual dictionaries

Our images were based on the crowdsourced bilingual dictionaries assembled by (Pavlick et al., 2014). Most of the bilingual dictionaries contain approximately 10,000 foreign words, but the exact number varies per language, since Pavlick et al filtered the dictionaries based on the estimated quality of the crowd workers making the contribution in order to discard poor translations. As a helpful guide, we have grouped the languages by ranges of word counts, so that readers can get a sense of how large the corpus for each language is.

- 8,000-10,000 words: Afrikaans, Albanian, Amharic, Arabic, Azerbaijani, Basque, Belarusian, Bengali, Bishnupriya Manipuri, Bosnian, Bulgarian, Catalan, Central Bicolano, Croatian, Danish, Dutch, Esperanto, Filipino, Finnish, French, Galician, German, Greek, Gujarati, Haitian, Hebrew, Hindi, Hungarian, Ilokano, Indonesian, Italian, Japanese, Javanese, Kannada, Kapampangan, Latvian, Lithuanian, Macedonian, Malay, Malayalam, Marathi, Nepali, Norwegian Nynorsk, Norwegian, Piedmontese, Polish, Portuguese, Punjabi, Romanian, Russian, Serbian, Serbo-Croatian, Slovak, Slovenian, Somali, Spanish, Sundanese, Swedish, Tamil, Telugu, Turkish, Ukrainian, Waray, Welsh, Urdu, Uzbek

- 5,000-8,000 words: Breton, Czech, Frisian, Georgian, Irish, Korean, Low Saxon,

Luxembourgish, Swahili, Uighur, Vietnamese

- 1,000-5,000 words: Aragonese, Armenian, Chinese, Neapolitan, Sicilian, Thai, Yoruba
- 100-1,000 words: Icelandic, Malagasy, Newar, Pashto, Persian, Sindhi
- <100 words: Ido, Kazakh, Kurdish, Wolof

A.2 Size of the corpus

We show a full breakdown of our corpus in Table 22. As a reminder, a same translation word (stw) refers to words whose translations are exact matches of the word itself, non-stw words show the count where that is not the case.

	Words in Dict		Before Filtering			After Filtering		
Language	All	Non-stw	Images	Web Pages	Webcrawl Words	Images	Web Pages	Webcrawl Words
Afrikaans	9,719	8,147	923,000	692,000	456,860,000	324,000	541,000	322,327,000
Albanian	9,600	9,036	923,671	665,880	439,126,000	324,000	520,000	309,815,000
Amharic	8,051	8,051	765,000	574,000	378,324,000	268,000	448,000	266,918,000
Arabic	9,813	9,811	941,011	685,467	421,236,351	329,826	542,544	360,831,818
Aragonese	3,141	2,557	298,000	224,000	147,783,000	105,000	175,000	104,265,000
Armenian	1,652	1,652	157,000	118,000	77,691,000	55,000	92,000	54,813,000
Asturian	3,486	2,535	331,000	248,000	163,828,000	116,000	194,000	115,585,000
Azerbaijani	9,942	9,287	931,691	665,843	439,126,000	327,000	520,000	309,815,000
Basque	9,686	8,728	920,000	690,000	455,171,000	323,000	539,000	321,135,000
Belarusian	10,087	9,973	958,000	719,000	474,594,000	336,000	562,000	334,838,000
Bengali	10,086	10,083	936,666	615,264	584,644,773	401,984	429,246	495,398,451
Bishnupriya Manipuri	9,874	9,871	938,000	704,000	464,460,000	329,000	550,000	327,689,000
Bosnian	10,022	9,425	974,803	765,833	504,995,000	342,000	598,000	356,287,000
Breton	6,895	2,285	655,000	491,000	324,278,000	230,000	384,000	228,786,000
Bulgarian	10,181	10,180	988,989	754,797	498,239,000	347,000	590,000	351,521,000
Catalan	9,999	9,046	950,000	713,000	470,371,000	333,000	557,000	331,860,000
Cebuano	8,510	6,547	791,476	558,039	350,153,208	278,000	441,184	32,841,835
Central Bicolano	9,935	6,406	944,000	708,000	466,993,000	331,000	553,000	329,476,000
Chinese	3,315	3,314	292,521	221,305	146,094,000	103,000	173,000	103,073,000
Croatian	9,903	9,301	941,000	706,000	466,149,000	330,000	552,000	328,881,000
Czech	7,392	6,960	702,000	527,000	347,923,000	246,000	412,000	245,469,000
Danish	8,376	6,998	796,000	597,000	393,524,000	279,000	466,000	277,642,000
Dutch	9,877	8,082	957,554	814,109	507,544,154	335,186	663,886	337,735,314
English	263,098	N/A	24,596,102	18,447,000	12,170,541,000	8,629,000	14,412,000	8,586,641,000
Esperanto	8,024	7,336	762,000	572,000	377,479,000	267,000	447,000	266,322,000
Filipino (Tagalog)	9,436	8,063	906,702	693,036	456,860,000	318,000	541,000	322,327,000
Finnish	10,018	8,700	952,000	714,000	471,216,000	334,000	558,000	332,455,000
French	9,887	8,166	962,222	816,834	613,624,883	374,849	673,147	451,973,804
Frisian	6,383	4,497	606,000	455,000	299,788,000	213,000	355,000	211,508,000
Galician	9,987	8,763	949,000	712,000	469,527,000	333,000	556,000	331,264,000
Georgian	5,315	5,314	505,000	379,000	249,964,000	177,000	296,000	176,356,000
German	9,807	8,175	953,052	826,086	522,788,572	381,520	662,193	406,915,283
Greek	9,899	9,897	940,000	705,000	465,304,000	330,000	551,000	328,285,000
Gujarati	9,979	9,975	945,875	305,097	200,985,000	332,000	238,000	141,800,000
Haitian	9,188	5,865	873,000	655,000	432,370,000	306,000	512,000	305,049,000
Hebrew	8,195	8,195	779,000	584,000	385,080,000	273,000	456,000	271,684,000
Hindi	9,150	9,147	889,789	626,673	413,792,000	312,000	490,000	291,941,000
Hungarian	9,850	9,020	958,540	756,537	499,083,000	336,000	591,000	352,117,000
Icelandic	822	738	78,000	59,000	38,846,000	27,000	46,000	27,407,000
Ido	68	48	6,000	5,000	3,378,000	2,000	4,000	2,383,000
Ilokano	9,333	4,449	887,000	665,000	439,126,000	311,000	520,000	309,815,000
Indonesian	9,773	7,683	946,444	834,041	467,016,410	269,457	612,703	299,680,811
Irish	7,301	6,334	694,000	521,000	343,700,000	243,000	407,000	242,490,000
Italian	9,518	8,310	927,027	814,854	579,321,695	363,685	666,641	449,642,224
Japanese	8,071	8,071	767,000	575,000	379,168,000	269,000	449,000	267,513,000
Javanese	9,877	7,575	938,000	704,000	464,460,000	329,000	550,000	327,689,000
Kannada	9,924	9,921	943,000	707,000	466,149,000	331,000	552,000	328,881,000
Kapampangan	9,870	3,646	938,000	704,000	464,460,000	329,000	550,000	327,689,000
Kazakh	30	30	3,000	2,000	1,689,000	1,000	2,000	1,192,000
Korean	7,435	7,434	706,000	530,000	349,612,000	248,000	414,000	246,660,000
Kurdish	33	33	3,000	2,000	1,689,000	1,000	2,000	1,192,000
Latvian	9,939	9,585	962,034	604,692	398,591,000	337,000	472,000	281,217,000
Lithuanian	9,939	7,741	944,000	708,000	466,993,000	331,000	553,000	329,476,000
Low Saxon	7,344	5,637	698,000	524,000	345,389,000	245,000	409,000	243,681,000
Luxembourgish	6,609	4,545	628,000	471,000	310,766,000	220,000	368,000	219,254,000
Macedonian	10,095	9,972	959,000	719,000	474,594,000	336,000	562,000	334,838,000
Malagasy	164	159	16,000	12,000	7,600,000	6,000	9,000	5,362,000
Malay	9,351	7,823	888,000	666,000	439,126,000	312,000	520,000	309,815,000
Malayalam	10,124	10,124	962,000	722,000	476,283,000	337,000	564,000	336,030,000
Marathi	9,988	9,987	949,000	712,000	469,527,000	333,000	556,000	331,264,000
Neapolitan	4,493	3,441	427,000	320,000	211,118,000	150,000	250,000	148,950,000
Nepali	9,916	9,915	700,479	396,109	260,942,000	246,000	309,000	184,102,000
Newar	262	262	25,000	19,000	12,667,000	9,000	15,000	8,937,000
Norwegian (Nynorsk)	8,473	6,976	805,000	604,000	398,591,000	282,000	472,000	281,217,000
Norwegian	9,083	7,603	863,000	647,000	426,459,000	303,000	505,000	300,878,000
Pashto	331	331	31,000	23,000	15,201,000	11,000	18,000	10,724,000
Persian (Farsi)	921	921	87,843	76,894	50,668,000	31,000	60,000	35,748,000
Piedmontese	9,294	7,308	883,000	662,000	436,592,000	310,000	517,000	308,028,000
Polish	9,764	9,159	928,000	696,000	459,393,000	326,000	544,000	324,114,000
Portuguese	9,873	8,695	938,000	704,000	464,460,000	329,000	550,000	327,689,000
Punjabi	9,827	9,827	934,000	701,000	462,771,000	328,000	548,000	326,497,000
Romanian	9,880	8,819	963,377	758,086	499,928,000	338,000	592,000	352,712,000
Russian	9,962	9,958	946,000	710,000	468,682,000	332,000	555,000	330,668,000
Serbian	10,146	10,039	979,731	764,291	504,150,000	344,000	597,000	355,691,000
Serbo-Croatian	10,057	9,590	955,000	716,000	472,060,000	335,000	559,000	333,051,000
Sicilian	1,751	1,491	166,000	125,000	82,758,000	58,000	98,000	58,388,000
Sindhi	36	36	3,000	2,000	1,689,000	1,000	2,000	1,192,000
Slovak	9,939	9,283	893,962	648,120	427,303,000	314,000	506,000	301,474,000
Slovenian	7,927	7,354	753,000	565,000	372,412,000	264,000	441,000	262,747,000
Somali	9,907	7,177	904,728	579,501	382,546,000	317,000	453,000	269,897,000
Spanish	9,825	8,778	959,099	699,244	450,079,676	305,749	553,368	380,362,877
Sundanese	9,909	4,726	941,000	706,000	466,149,000	330,000	552,000	328,881,000
Swahili	7,019	6,132	666,805	500,104	330,189,000	234,000	391,000	232,957,000
Swedish	9,551	8,086	928,935	720,860	475,438,000	326,000	563,000	335,434,000
Tamil	9,449	9,448	901,355	511,136	336,945,000	316,000	399,000	237,723,000
Telugu	9,751	9,751	933,566	364,407	240,675,000	328,000	285,000	169,802,000
Thai	4,487	4,487	423,768	334,513	220,407,000	149,000	261,000	155,503,000
Turkish	10,007	9,263	984,243	781,376	483,687,039	372,415	609,274	439,236,149
Uighur	5,650	5,650	495,402	176,736	104,323,748	174,000	122,660	54,723,339
Ukrainian	10,027	9,990	972,517	720,864	475,438,000	341,000	563,000	335,434,000
Urdu	9,999	9,998	930,003	625,048	412,103,000	326,000	488,000	290,749,000
Uzbek	9,696	5,630	904,713	579,421	326,562,177	188,211	430,682	108,250,659
Vietnamese	5,911	4,586	558,470	494,680	325,966,000	196,000	386,000	229,978,000
Waray	8,489	5,368	806,000	605,000	399,436,000	283,000	473,000	281,812,000
Welsh	9,923	7,272	917,093	562,247	370,724,000	322,000	439,000	261,555,000
Wolof	45	36	4,000	3,000	1,689,000	1,000	2,000	1,192,000
Yoruba	1,802	1,523	140,134	92,334	60,802,000	49,000	72,000	42,897,000

Table 22: Statistics about the number of images and words in our data set. Estimated numbers are rounded to the nearest 1,000.

A.3 Package report

We show a full example of a package report in Figure 58.

Figure 58: Example package report for French


```
{
"total_file_size": "250.2 GB",
"avg_file_size": "260.0 kB",
"total_words": 9887,
"extension_counts": {
"other": 1343,
"bmp": 471,
"jpg": 816925,
"png": 108234,
"gif": 33922,
"svg": 1171,
"ico": 156
},
"avg_width": 858,
"min_images_per_word": 20,
"median_images_per_word": 98,
"top_10_hostname_counts": {
"fr.dreamstime.com": 10813,
"www.youtube.com": 14041,
"www.pinterest.com": 5833,
"fr.wikipedia.org": 17740,
"en.wikipedia.org": 19294,
"fr.123rf.com": 6985,
"www.purepeople.com": 5484,
"www.ouest-france.fr": 7154,
"twitter.com": 10906,
"commons.wikimedia.org": 6520
},
"num_unique_hosts": 246610,
"max_images_per_word": 100,
"avg_height": 662,
"total_images": 962222
}
```


A.4 Visualizations

In this section, we will quickly walk through the process we used to generate the visualizations of SIFT, Histogram, and AlexNet features.

SIFT

In Figure 22, we show matching features from two grayscale images, an original one on the left and a rotated and scaled up one on the right. For this visualization, we used the Computer Vision System Toolbox in MATLAB. More specifically, we used the examples here: https://www.mathworks.com/help/vision/ref/showmatchedfeatures.html

We combined the approaches in both examples, and used SURF features (Bay et al., 2006), which at a high level are an improved version of SIFT. After reading in our original image and creating a rotated and resized version, we detected and extracted the SURF features in both images. We then match the features between the two images, plotting the images side by side with the matching points highlighted and connected.

We show the code in Figure 59. Note: Figure 38 uses this same approach except with two distinct original images.

Figure 59: MATLAB code snippet for matching features

```
I1 = imread('original-image-file');
I2 = imresize(imrotate(I1,-20), 1.2);
points1 = detectSURFFeatures(I1);
points2 = detectSURFFeatures(I2);
[f1, vpts1] = extractFeatures(I1, points1);
[f2, vpts2] = extractFeatures(I2, points2);
indexPairs = matchFeatures(f1, f2) ;
matchedPoints1 = vpts1(indexPairs(:, 1));
matchedPoints2 = vpts2(indexPairs(:, 2));
figure; ax = axes;
showMatchedFeatures(I1,I2,matchedPoints1,matchedPoints2,'
    montage','Parent',ax);
legend(ax, 'Matches - original','Matches - rotate+resize');
```

In Figure 24, we show a grid of five images with the strongest features identified. Once again, we used the Computer Vision System Toolbox in MATLAB. More specifically, we used the example here: https://www.mathworks.com/help/vision/ref/surfpoints.plot.html

We first loaded each image, detected and extracted SURF features for it. We then selected the strongest N points (where N is determined by experimentation) and plotted them on top of the image. We show the code in Figure 60.

Figure 60: MATLAB code snippet for strongest features

```
I = imread('original-image-file');
N = <number of points desired>;
points = detectSURFFeatures(I);
[features, valid_points] = extractFeatures(I,points);
imshow(I);
hold on;
strongestPoints = valid_points.selectStrongest(N);
strongestPoints.plot('showOrientation',true);
```

Histogram

For Figures 27 and 28, we used MATLAB's histogram plot and colormap to plot the occurrences of each R, G, and B channel. We show the code in Figure 61.

Figure 61: MATLAB code snippet for strongest features

```
input = double(imread('original-image-file'));
hist(reshape(input,[],3),1:max(input(:)));
colormap([1 0 0; 0 1 0; 0 0 1]);
xlabel('Color intensity')
ylabel('Occurrence count')
```

AlexNet

For Figures 32, 33, 34, 35 and 36, we show the original image next to the strongest activating feature for the specified convolutional layer, when the image is fed forward through the AlexNet. For this visualization, we used the Neural Network Toolbox, Image Processing Toolbox and the Neural Network Toolbox Model for AlexNet Network in MATLAB. More specifically, we followed the examples here: http://www.mathworks.com/help/nnet/examples/visualize-activations-of-a-convolutional-neural-network.html

We show the code in Figure 62. To get results for layer N, just swap conv1 for convN.

Figure 62: MATLAB code snippet for AlexNet layer 1

```
net = alexnet;
im = imread('original-image-file');
imgSize = size(im);
imgSize = imgSize(1:2);
act1 = activations(net,im,'conv1','OutputAs','channels
    ');
sz = size(act1);
act1 = reshape(act1,[sz(1) sz(2) 1 sz(3)]);
[maxValue,maxValueIndex] = max(max(max(act1)));
act1chMax = act1(:,:,:,maxValueIndex);
act1chMax = mat2gray(act1chMax);
act1chMax = imresize(act1chMax,imgSize);
imshowpair(im,act1chMax,'montage')
```

For the example in Figure 37, we constructed what a deep dream image (Google, 2015) that strongly activates the channel of the network's layers that relates to Siamese cats (285). This allows us to highlight the image features that were learned by the network for a particular channel. AlexNet has 1,000 channels and each one relates via

ImageNet to a specific thing, like *Siamese cat*, *African elephant*, or *strawberry*.

Deep Dream is a feature visualization technique in deep learning that synthesizes images that strongly activate network layers. By visualizing these images, you can highlight the image features learned by a network. These images are useful for understanding and diagnosing network behavior.

Figure 63: MATLAB code snippet for deep dream images with AlexNet

```
net = alexnet;
layer = 20;
channels = 285;
I = deepDreamImage(net,layer,channels, ...
'Verbose',false, ...
'NumIterations',50);
figure
montage(I)
name = net.Layers(layer).Name;
title(['Layer ',name,' Features'])
```

References

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. 2006. Surf: Speeded up robust features. *Computer vision–ECCV 2006* pages 404–417.

Shane Bergsma and Randy Goebel. 2011. Using visual information to predict lexical preference. In *RANLP*. pages 399–405.

Shane Bergsma and Benjamin Van Durme. 2011. Learning bilingual lexicons using the visual similarity of labeled web images. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Anna Bosch, Andrew Zisserman, and Xavier Munoz. 2007. Image classification using random forests and ferns. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, pages 1–8.

Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. 1990. A statistical approach to machine translation. *Computational linguistics* 16(2):79–85.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known English word lemmas. *Behavior research methods* 46(3):904–911.

David L Chen and William B Dolan. 2011. Building a persistent workforce on mechanical turk for multilingual data collection. In *Proceedings of The 3rd Human Computation Workshop (HCOMP 2011), August*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, pages 248–255.

Thomas Deselaers, Daniel Keysers, and Hermann Ney. 2008. Features for image retrieval: an experimental comparison. *Information retrieval* 11(2):77–107.

Anita L. Verő Douwe Kiela and Stephen Clark. 2016. Comparing Data Sources and Architectures for Deep Visual Representation Learning in Semantics. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-16)*.

Facebook. 2008. 10 billion photos. https://www.facebook.com/notes/facebook-engineering/10-billion-photos/30695603919/.

Robert Fergus, Li Fei-Fei, Pietro Perona, and Andrew Zisserman. 2005. Learning object categories from google's image search. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*. IEEE, volume 2, pages 1816–1823.

Francis Ferraro, Nasrin Mostafazadeh, Ting-Hao Huang, Lucy Vanderwende, Jacob Devlin, Michel Galley, and Margaret Mitchell. 2015. A survey of current datasets for vision and language research. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 207–213. http://aclweb.org/anthology/D15-1021.

Pascale Fung and Lo Yuen Yee. 1998. An IR approach for translating new words from nonparallel, comparable texts. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, pages 414–420.

Eric Gaussier, J-M Renders, Irina Matveeva, Cyril Goutte, and Hervé Déjean. 2004. A geometric view on bilingual lexicon extraction from comparable corpora. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, page 526.

Google. 2010. Ooh! Ahh! Google Images presents a nicer way to surf the visual web. https://googleblog.blogspot.com/2010/07/ooh-ahh-google-images-presents-nicer.html.

Google. 2015. Inceptionism: Going deeper into neural networks. https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html.

Google. 2017a. Google custom search api reference. https://developers.google.com/custom-search/json-api/v1/reference/cse/list.

Google. 2017b. Google search request format - language filters. https://www.google.com/support/enterprise/static/gsa/docs/admin/72/gsa_doc_set/xml_-reference/request_format.html#1077312.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *ACL*. volume 2008, pages 771–779.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 770–778.

Julian Hitschler, Shigehiko Schamoni, and Stefan Riezler. 2016. Multimodal pivots for image caption translation. *arXiv preprint arXiv:1601.03916* .

LLC ImageMagick Studio. 2008. Imagemagick. https://www.imagemagick.org/.

Ann Irvine and Chris Callison-Burch. 2017. A comprehensive analysis of bilingual lexicon induction. *Computational Linguistics* .

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, pages 675–678.

Douwe Kiela. 2016. Mmfeat: A toolkit for extracting multi-modal features. In *Proceedings of ACL*.

Douwe Kiela, Felix Hill, Anna Korhonen, and Stephen Clark. 2014. Improving multi-modal representations using image dispersion: Why less is sometimes more. In *ACL (2)*. pages 835–841.

Douwe Kiela, Ivan Vulić, and Stephen Clark. 2015. Visual bilingual lexicon induction with transferred convnet features. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 148–158. http://aclweb.org/anthology/D15-1015.

Adam Kilgarriff. 2007. Googleology is bad science. *Computational linguistics* 33(1):147–151.

Alexandre Klementiev and Dan Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 817–824.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proceedings of the ACL-02 workshop on Unsupervised lexical acquisition-Volume 9*. Association for Computational Linguistics, pages 9–16.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. pages 1097–1105.

Victor Lavrenko, Martin Choquette, and W Bruce Croft. 2002. Cross-lingual relevance models. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 175–182.

Gina-Anne Levow, Douglas W Oard, and Philip Resnik. 2005. Dictionary-based techniques for cross-language information retrieval. *Information processing & management* 41(3):523–547.

David G Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision* 60(2):91–110.

Douglas L Nelson, Cathy L McEvoy, and Thomas A Schreiber. 2004. The University of South Florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers* 36(3):402–407.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics* 29(1):19–51.

Ellie Pavlick, Matt Post, Ann Irvine, Dmitry Kachaev, and Chris Callison-Burch. 2014. The language demographics of Amazon Mechanical Turk. *Transactions of the Association for Computational Linguistics* 2:79–92.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated English and German corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. Association for Computational Linguistics, pages 519–526.

Charles Schafer and David Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *proceedings of the 6th conference on Natural language learning-Volume 20*. Association for Computational Linguistics, pages 1–7.

David Sculley. 2010. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*. ACM, pages 1177–1178.

Selenium. 2006. Selenium webdriver. http://www.seleniumhq.org/projects/webdriver/.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. pages 806–813.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* .

Dick Sites. 2013. Compact language detection 2. https://github.com/CLD2Owners/cld2.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 1–9.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2012. Bilingual lexicon extraction from comparable corpora using label propagation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 24–36.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research* 37:141–188.

Andrea Vedaldi and Brian Fulkerson. 2010. Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the 18th ACM international conference on Multimedia*. ACM, pages 1469–1472.

Luis Von Ahn and Laura Dabbish. 2004. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, pages 319–326.

Ivan Vulić, Douwe Kiela, Stephen Clark, and Marie-Francine Moens. 2016. Multi-modal representations for improved bilingual lexicon learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 188–194. http://anthology.aclweb.org/P16-2031.

Ivan Vulic and Marie-Francine Moens. 2013. Cross-lingual semantic similarity of words as the similarity of their semantic word responses. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*. ACL, pages 106–116.