Scaling Hierarchical Coreference with Homomorphic Compression

Michael Wick MICHAEL.WICK@ORACLE.COM
Oracle Labs,
Burlington, MA

Swetasudha Panda SWETASUDHA.PANDA@ORACLE.COM
Orable Labs,
Burlington, MA

Joseph Tassarotti JTASSARO@MIT.EDU
Massachusetts Institute of Technology
Cambridge, MA

Jean-Baptiste Tristan JEAN.BAPTISTE.TRISTAN@ORACLE.COM
Orable Labs,
Burlington, MA

Abstract

Locality sensitive hashing schemes such as simhash provide compact representations of multisets from which similarity can be estimated. However, in certain applications, we need to estimate the similarity of dynamically changing sets. In this case, we need the representation to be a homomorphism so that the hash of unions and differences of sets can be computed directly from the hashes of operands. We propose two representations that have this property for cosine similarity (an extension of simhash and angle-preserving random projections), and make substantial progress on a third representation for Jaccard similarity (an extension of minhash). We employ these hashes to compress the sufficient statistics of a conditional random field (CRF) coreference model and study how this compression affects our ability to compute similarities as entities are split and merged during inference. We also provide novel statistical analysis of simhash to help justify it as an estimator inside a CRF, showing that the bias and variance reduce quickly with the number of hash bits. On a problem of author coreference, we find that our simhash scheme allows scaling the hierarchical coreference algorithm by an order of magnitude without degrading its statistical performance or the model's coreference accuracy, as long as we employ at least 128 or 256 bit hashes. Angle-preserving random projections further improve the coreference quality, potentially allowing even fewer dimensions to be used.

1. Introduction

Probabilistic models in machine learning, such as conditional random fields (CRFs), are widely successful at modeling many problems at the heart of knowledge base construction, including those in natural language processing, information extraction and data integration. However, when dealing with natural language data, the underlying feature representations are often sparse, high-dimensional and dynamic (*i.e.,* they change during inference). In this paper we consider the task of coreference resolution, in which the goal is to partition a set of mentions into the entities to which they refer. We might represent each mention with a

feature vector in which each dimension corresponds to a word or n-gram. Since only a small subset of the vocabulary is observed per mention, most elements of the vector are zero.

Given the model and these representations, inference entails making decisions about whether two entities should be coreferent. To make such decisions, the model estimates a probability that involves computing the similarities between the aggregate feature representations of the two entities' mentions. Since *all* the vectors are both sparse and high-dimensional, these similarity operations are computationally expensive because the sparse data structures supplant the dense arrays that would otherwise support fast look-ups. Moreover, as the inference algorithm makes decisions about whether or not two entities are coreferent, we may have to split or merge the entities and thus we must *update the feature vector* to reflect these changes. Maintaining such sparse-vector representations in the inner-loop of probabilistic inference is expensive, especially as the entities grow in size.

In order to cope with the computational problems associated with sparse, high dimensional dynamic feature representations, we propose using *homomorphic compression*, in which the compressed representations of intermediate inference results can be computed directly from their operands, allowing inference to run directly on the compressed representations of the data even as they change. In this paper, we consider several such schemes to scale hierarchical coreference. First, we propose a novel homomorphic cosine-preserving hashing scheme based on simhash [Charikar, 2002] that also supports addition and subtraction to more efficiently represent the data and the evolving intermediate results of probabilistic inference. Second, because various linear norm-preserving random projections also preserve angles [Magen, 2002], we can directly compute cosine similarity on projected data – linearity of the projections means that they too can be updated dynamically. The resulting angle estimates are superior to the homomorphic simhash representation, at the cost of reduced efficiency for certain operations. Third, we develop a homomorphic version of minhash [Broder, 1997b] to support Jaccard similarity. Our current algorithm is biased, but the bias appears small in practice for the situations we have considered. Although the minhash based set representations are not currently employed in hierarchical coreference, they might be useful in other models or applications that involve binary features over changing sets [Culotta et al., 2007b].

We provide error analysis for all three schemes, collating and extending known results, and in the case of simhash, we provide novel statistical analysis for its use as a direct estimator for $\cos(\theta)$ that shows the bias and variance decrease rapidly with the number of bits, helping to justify its use in a CRF for a task like coreference. On a hierarchical model for coreference resolution, the proposed simhash scheme improves the speed of probabilistic inference by an order of magnitude while having little effect on model quality. Moreover, we find that the underlying random projection representation can provide even better cosine estimates than simhash, at the cost of not being able to use certain fast bitwise-operations to compute similarities. Finally, we briefly evaluate homomorphic minhash and show that even though there are possible pathological cases, as long as we employ enough hash functions, the estimates are reasonably close to the true Jaccard, albeit, biased.

2. Coreference Resolution: Background and Challenges at Scale

Coreference resolution Coreference resolution is the problem of determining whether different *mentions* refer to the same underlying *entity* [Getoor and Machanavajjhala, 2012].

For example, in the sentence "In a few years [McDaniels] would replace [Belichick] as the forty-two year old [quarterback], [Tom Brady] retires." coreference must correctly determine that "quarterback" refers to Tom Brady and not Belichick or McDaniels. This type of coreference is sometimes referred to as noun-phrase or within document coreference and often relies upon linguistic and discourse motivated features [Raghunathan et al., 2010]. Coreference resolution arises in many other situations; for example, when merging two or more databases together it is desirable to remove duplicates that result from the merge, a problem sometimes termed record linkage or deduplication [Newcombe et al., 1959]. Coreference is also foundational to *knowledge base construction* which requires that we combine information about entities of interest from multiple sources that might mention them in different contexts. For example, if we were to build a knowledge base of all scientists in the world — similar to Google scholar — we would need to perform the task of *author coreference* to determine who authored what papers [Han et al., 2004, Culotta et al., 2007a]. Are the following two mentions of "J Smith" the same author?

> *V Khachatryan, AM Sirunyan,...,* **J Smith.** *Observation of the diphoton decay of the Higgs boson and measurement of its properties. The European Physical Journal 2014*
>
> *S Chatrchyan, V Khachatryan, AM Sirunyan, A Tumasyan, W Adam,* **J Smith**. *Jet production rates in association with W and Z bosons in pp collisions. J High Energy Physics 2012*

Although generally this is a difficult problem, it can be solved with machine learning since features of the mentions such as the words in the title (both have "Boson" in common), the topic of the title (both are about a similar subfield of physics), the journal (both are physics journals) and the co-authors (there appears to be a co-author in common) provide some evidence about whether or not the two "J Smith's" might be the same person.

In order to solve the problem, it is thus common to extract such contextual features about each mention, such as in the above example, features from the title, co-author list, venue, year and author-name and employ them in a probabilistic model. These features are typically the raw words, character-ngrams and normalized variants thereof, sometimes with positive real-valued weights to indicate the importance (e.g., via TFIDF) of each feature. Then, given such features, a coreference model measures the similarities between mentions via functions such as cosine-similarity. In contrast to within document coreference discussed earlier, this type of coreference resolution problem is better suited for similarity based models, such as the ones we will use in the following. Moreover, since coreference decisions are not restricted by document-boundaries, the entities can grow unbounded in size, making compact representations of their growing feature sets especially important.

Typically, the model is a discriminative conditional random field (CRF) that measure the probability of an assignment of mentions to entities conditioned on the observed features [McCallum and Wellner, 2003]. The model factorizes into potentials that score local coreference decisions. Local search procedures such as greedy-agglomerative clustering or Markov-chain Monte Carlo (MCMC) find the most likely assignment of mentions to entities [McCallum and Wellner, 2003, Culotta et al., 2007a, Wick et al., 2012].

In *pairwise* models, potential functions measure the compatibility of two mentions being in the same cluster. An issue with such models is that the possible pairwise comparisons scales

quadratically with the number of mentions. An alternative class of models that avoids this quadratic blow-up are *entity-based*, in which entities are treated as first-class variables with their own set of inferred features, and potentials measure compatibility between mentions and entities. However, entity-based models come with their own scalability challenges. To illustrate the problem (and our solution), we focus on an entity-based model called *hierarchical coreference*, which recently won a challenge to disambiguate inventor names for the USPTO, due to its accuracy and scalability [Uni, 2016, Monath and McCallum, 2016].

Hierarchical Coreference In the hierarchical coreference model, mentions are organized into latent tree structures [Wick et al., 2012]. There is one tree per entity with mentions at the leaves and intermediate nodes as "subentities" that organize subsets of the entity's mentions. Rather than modeling interactions between mention-pairs, the potential functions measure compatibility between child and parent nodes in the tree. The score of a given assignment of mentions into latent trees is the product of all model potentials which includes these child-parent compatibility scores as well as some additional priors on tree-shape and entities. These compatibility scores are parametrized cosine functions.

Each mention is endowed with multiple feature variables that each capture a subset of the total features. For example, in author coreference, one feature variable might capture features of the author's name and another might capture features of the title and venue of the paper. Colloquially, we refer to these feature variables as "bags" since they inherit the usual bags-of-words assumption. We distinguish between different "bag-types" which each capture different types of features (e.g., the author name bag, title words bag, co-author bag, etc). The other nodes in the tree also contain feature variables (bags), but the values of these variables are determined by the current assignment of children to that parent node. In particular, *a parent's bag is the sum of all its children's bags.* In order to maintain this invariant, the bag representations must be updated to reflect the current state of inference, and hence for efficiency reasons, representations must be homomorphic with respect to operations that will be performed on the bags.

Interpreting bags as vectors, the cosine distance between them is used to calculate their compatibility. The primary potential functions measure the compatibility between each child's bag and its parent's bag. There is one potential for each bag-type. For example, to measure the compatibility between a node z_i and z_j, let y_{ij} be the binary variable that is 1 if and only if z_j is the parent of z_i, and let $b_i^{(0)}$ and $b_j^{(0)}$ be a bag for z_i and z_j respectively, then the potential $\psi^{(0)}$ for "bag 0" scores a coreference decision as:

$$\psi^{(0)}(z_i, z_j, y_{ij}; w, t) = \begin{cases} 1 & y_{ij} = 0 \\ \exp\left(w(\cos(b_i^{(0)}, b_j^{(0)} - b_i^{(0)}) - t)\right) & y_{ij} = 1 \end{cases} \quad (1)$$

where w is a real-valued weight and t is a real-valued translation parameter for potential $\psi^{(0)}$. The potentials for each bag-type have parameters w, t that we can fit to data.

Because only a small handful of features are ever observed for a given mention, typical implementations of hierarchical coreference employ sparse-vector representations to represent bags (*e.g.,* the implementation found in FACTORIE [McCallum et al., 2008, 2009]).

However, a key disadvantage of sparse vector representations is that they must store the indices and weights of the non-zero elements, which means that the data-structures

must dynamically change in size as MCMC splits and merges entities. As the sizes of the entities grow, these operations become increasingly expensive. Similar issues arise in other entity-based models where features of entities are aggregated from those of mentions. Thus, while entity-based models avoid the quadratic comparison issue of pairwise models, a straight-forward sparse representation of their feature vectors is no longer efficient.

Is there an alternative representation of feature vectors which (a) allows fast evaluation of cosine-similarity, and (b) can be efficiently dynamically updated? As we describe in the next section, the simhash hashing function [Charikar, 2002] provides a representation with property (a). However, the standard simhash cannot be updated as vectors are modified. We therefore develop a variant which we call *homomorphic* simhash, which has both properties (a) and (b). We also identify two other schemes that support these properties.

3. Homomorphic Representations for Measuring Similarity

We now discuss three different homomorphic representations that support addition and subtraction in the compressed space, while preserving similarity estimates. We propose homomorphic simhash and a related random projection for cosine similarity of high-dimensional vectors and multisets; and homomorphic minhash for Jacard similarity of sets.

3.1 Homomorphic simhash and its Statistical Properties

Background: simhash A *locality-sensitive hash function* for a distance metric d on a set of objects S is a function H such that given $x, y \in S$, we can estimate $d(x, y)$ from the hashed representations $H(x)$ and $H(y)$. Simhash [Charikar, 2002] is a locality sensitive hash function for cosine similarity.

To understand simhash, it is first helpful to consider the following randomized process: Imagine that we have two vectors a and b on the unit hypersphere in the Euclidean space $\mathcal{R}^d$ with angle θ between them, and we want to produce an estimate of $\cos(\theta)$. Select a random d-dimensional vector u by sampling each of its coordinates independently from $N(0, 1)$. Let the random variable X have value 1 if a and b are on different sides of the hyperplane orthogonal to u, and 0 otherwise. Then X is a Bernoulli random variable with expectation:

$$\mathbb{E}[X] = 1 - \mathbb{P}(sign(a \cdot u) = sign(b \cdot u)) \tag{2}$$

$$= 1 - \frac{\theta}{\pi} \tag{3}$$

Let $X_1, ..., X_n$ be the result of independently repeating this process several times, and set $\overline{X}_n = \frac{1}{n} \sum_{i=1}^{n} X_i$. Then $\mathbb{E}[\overline{X}_n] = 1 - \frac{\theta}{\pi}$, and hence $\mathbb{E}[\pi(1 - \overline{X}_n)] = \theta$, so that we can use[1] $\cos(\pi(1 - \overline{X}_n))$ as an estimator of $\cos(\theta)$.

The idea behind simhash is to come up with a hash representation which lets us reproduce this randomized estimator: to construct a function H which produces n-bit hashes, we first randomly sample n vectors $u_1, \ldots, u_n$ as above. Then, given a vector a, the hash $H(a)$ is the length n bit sequence in which the ith bit is 1 if the sign of $a \cdot u_i$ is positive and 0

1. Charikar [Charikar, 2002] notes that for some applications, $1 - \frac{\theta}{\pi}$ may be a good enough approximation of $\cos(\theta)$, so that one can use $\overline{X}_n$ directly as an estimator of $\cos(\theta)$.

otherwise. Now, from two hashed representations $H(a)$ and $H(b)$, if the ith bit in the two hashes disagree, this is equivalent to X_i in the randomized process above being equal to 1. Thus, by counting the number of positions where the hashes are distinct and dividing by n, we get $\overline{X}_n$, thereby producing an estimate of $\cos(\theta)$.

Rather than constructing the hash function H by sampling the $u_1, \ldots, u_n$ vectors from the d-dimensional unit sphere uniformly, a common optimization is to instead sample them from $\{-1, 1\}^d$. This has two advantages. First, it is faster to compute the dot product since no floating point multiplication is involved. Second, rather than having to explicitly sample and store each u_i as a vector, we can replace each u_i by a 1-bit feature hash function h_i: the "value" of the vector represented by h_i is 1 at coordinate j if $h_i(j) = 1$ and is -1 if $h_i(j) = 0$. We write $a \cdot h_i$ for the dot product of a with the vector corresponding to h_i.

By restricting only to test vectors with coordinates of the form 1 and -1, the corresponding expected value of $\pi(1 - \overline{X}_n)$ is no longer exactly θ (see [Leskovec et al., 2014, Section 3.7.3] for an example), but for high-dimensional spaces, this approximation is known to be effective in practice [Henzinger, 2006].

Homomorphic simhash If we want to use simhash as a representation of feature vectors for entities in coreference resolution, then we need to be able to update the simhash representation as entities are merged and split. In particular, if we join nodes with feature vectors a and b, then the vector of their new parent will be $a + b$. However, if we only store $H(a)$ and $H(b)$, rather than the vectors a and b themselves, we cannot compute $H(a + b)$: the ith bit of $H(a)$ and $H(b)$ just records the sign of $a \cdot h_i$ and $b \cdot h_i$, and if these are different, we do not know what the sign of $(a + b) \cdot h_i$ should be. A similar problem occurs when we split a child with vector b from a parent with vector a, since the updated parent's hash should be $H(a - b)$.

Our solution is instead to store the actual dot product of $a \cdot h_i$ in the hash of a, rather than just the sign. That is, $H(a)$ is now an array of length n instead of an n-bit sequence. And since

$$(a + b) \cdot h_i = a \cdot h_i + b \cdot h_i \qquad \text{and} \qquad (a - b) \cdot h_i = a \cdot h_i - b \cdot h_i$$

we can compute $H(a + b)$ by adding component-wise the arrays for $H(a)$ and $H(b)$, and similarly for $H(a - b)$. Finally, we can still efficiently compute the cosine distance between two vectors a and b by examining the signs of the entries of $H(a)$ and $H(b)$. We call this representation *homomorphic* because H is a homomorphism with respect to the additive group structure on vectors.

Of course, storing each dot product instead of just the signs increases the size of our hashes. However, they are still small compared to the feature vectors and, more importantly, their sizes are fixed. We can also store both the dot product and the signs as a bit vector, making sure to update the sign vector after each operation based on the dot product. By storing the sign vector separately we can quickly count the signs in common between two vectors using bitwise operations.

Statistical Properties Recall that since $\mathbb{E}[\pi(1 - \overline{X}_n)] = \theta$, we can derive a plausible estimate of $\cos(\theta)$ from $\overline{X}_n$. In particular, let $g(x) = \cos(\pi(1-x))$ and consider the estimator $C_n = g(\overline{X}_n)$. We now describe some statistical properties of C_n. Our emphasis here is somewhat different from the standard analyses found in related work. The reason is that

LSHs like simhash are most commonly used for duplicate detection [Henzinger, 2006] and for approximate nearest neighbor search [Leskovec et al., 2014], which is quite different from our use case. In those settings, one wants to show that if two items x and y are very similar, then the distance estimated from $h(x)$ and $h(y)$ will very likely be quite small, and conversely if x and y are very different, then their estimated distances will be large. In such cases, the linear approximation to cosine $\overline{X}_n$ is sufficient. However, since we want to use the cosine distance estimates C_n as part of the potential function in our CRF, we are interested in additional statistical properties of the estimator.

Lemma 3.1. *C_n is consistent. In particular, $C_n \xrightarrow{a.s.} \cos(\theta)$.*

Proof. By the strong law of large numbers, we have that $\overline{X}_n \xrightarrow{\text{a.s.}} 1 - \frac{\theta}{\pi}$. Since g is continuous, by the continuous mapping theorem [Van Der Vaart, 1998],

$$g(\overline{X}_n) \xrightarrow{\text{a.s.}} g(1 - \frac{\theta}{\pi}) = \cos(\theta)$$

□

Lemma 3.2. $\mathbb{E}[C_n] = \cos(\theta) + E_n$, *where* $|E_n| \leq \frac{\pi^2}{8n}$

Proof Sketch. Set $\mu = \mathbb{E}[\overline{X}_n] = 1 - \frac{\theta}{\pi}$. The first degree Taylor series for $g(x)$ about μ is:

$$g(x) = \cos(\theta) + \pi \sin(\theta)(x - \mu) + R(x)$$

where R is the remainder term. We have then that:

$$\mathbb{E}[g(\overline{X}_n)] = \cos(\theta) + \pi \sin(\theta)(\mathbb{E}[\overline{X}_n] - \mu) + \mathbb{E}[R(\overline{X}_n)] \tag{4}$$
$$= \cos(\theta) + \mathbb{E}[R(\overline{X}_n)] \tag{5}$$

Thus it suffices to bound $|\mathbb{E}[R(\overline{X}_n)]|$, which can be done using Lagrange's remainder formula (see appendix). □

Lemma 3.3. $\mathbb{V}[C_n] = \frac{\pi^2 \sin^2(\theta)}{n} \cdot \frac{\theta}{\pi}(1 - \frac{\theta}{\pi}) + O(n^{-3/2})$

Proof Sketch. For intuition, note that the Taylor series above for g shows us that $g(x) \approx \cos(\theta) + \pi \sin(\theta)(x - \mu)$. So, recalling that $\mathbb{V}[C_n] = \mathbb{V}[g(\overline{X}_n)] = \mathbb{E}[g(\overline{X}_n)^2] - \mathbb{E}[g(\overline{X}_n)]^2$, and plugging in the approximation we get:

$$\mathbb{V}[g(\overline{X}_n)] \approx \mathbb{E}[\big(\cos(\theta) + \pi \sin(\theta)(\overline{X}_n - \mu)\big)^2] - \cos(\theta)^2 \tag{6}$$
$$= 2\pi \sin(\theta)\mathbb{E}[\overline{X}_n - \mu] + \pi^2 \sin^2(\theta)\mathbb{E}[(\overline{X}_n - \mu)^2] \tag{7}$$
$$= \pi^2 \sin^2(\theta)\mathbb{E}[(\overline{X}_n - \mu)^2] \tag{8}$$

To obtain the actual error bound, we carry out the same process but without dropping $R(x)$ from the Taylor approximation for g, and then once again use Lagrange's remainder formula to bound the remainder (see appendix). □

Finally, since C_n is equal to a Lipschitz continuous function of n independent Bernoulli random variables, we can use the the method of bounded differences [Dubhashi and Panconesi, 2009, Corollary 5.2], to derive the following Chernoff-like tail bound:

Lemma 3.4. $\mathbb{P}(|C_n - \cos(\theta)| > \delta + \frac{\pi^2}{8n}) \leq 2e^{-2n\delta^2/\pi^2}$.

3.2 Angle Preseving Random Projections for Cosine

The statistics underlying simhash, whether the hyperplanes are drawn from Gaussians or the d-dimensional hypercube (i.e., Rademacher distributions), are actually random projections for which the Johnson-Lidenstrauss lemma applies [Johnson and Lindenstrauss, 1982, Indyk and Motwani, 1998, Achlioptas, 2003]. The lemma states that any set of m points in d-dimensional Euclidean space can be embedded into n-dimensional Euclidean space, where n is logarithmic in m and independent of $d, n = O(\epsilon^{-2} \log m)$, such that all pairwise distances are maintained within an arbitrarily small factor $1 \pm \epsilon$, where $0 < \epsilon < 1$. Since the projections are linear, they are homomorphic with respect to addition and subtraction. Moreover, previous work shows that the norm-preserving property of a linear random projection A implies an angle-preserving property [Magen, 2002]; that is, for vectors u, v, let $\theta = \sphericalangle(v, u)$ and $\hat{\theta} = \sphericalangle(Av, Au)$. The result is the following:

$$\theta - \hat{\theta} \leq 2\sqrt{\epsilon} \text{ and } \frac{1}{1+\epsilon'} \leq \hat{\theta}/\theta \leq 1 + \epsilon' \tag{9}$$

where $\epsilon \leq \frac{1}{3}, \epsilon' = \frac{8}{\pi}\sqrt{\epsilon}$ and $n \geq 60\epsilon^{-2} \log m$. Therefore, we are justified in using the statistics underlying simhash to directly estimate the cosine similarity; viz., $\cos\theta \approx \cos\hat{\theta}$. More precisely, using a Taylor expansion, $|\cos\theta - \cos\hat{\theta}| \leq |2\sqrt{\epsilon}\sin\hat{\theta} + \frac{(2\sqrt{\epsilon})^2}{2}\cos\hat{\theta}| \leq 2\sqrt{\epsilon(1+\epsilon)}$. Although we lose the bit-representation that supports the fast bit-operations that makes simhash so efficient in applications such as webpage deduplication that employ the linear estimator, we potentially gain an improvement in the quality of the cosine estimate. Certainly, the estimate will be smoother since simhash essentially quantizes the angles into a finite number of possibilities equal to the number of bits. Since the same representation supports both types of estimates, we could even alternate between the estimation strategies as dictated by the particular situation: if a large number of similarity comparisons are performed for each entity then simhash makes more sense, otherwise a direct computation of cosine makes more sense. Regardless, as we will later see, both schemes provide sufficiently fast and accurate cosine estimates for coreference resolution.

3.3 Homomorphic Minhash for Jaccard Similarity

Minhash [Broder, 1997a, Broder et al., 2000] is a locality-sensitive hash function for Jaccard similarity, defined for binary vectors (encoding sets). The Jaccard similarity between two sets $S_1, S_2 \in \Omega$ is $J = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$. Minhash applies a random permutation (which in practice is accomplished using a random hash function, such as seeded 64-bit murmur hash) $\pi : \Omega \to \Omega$, on a given set $S \subset \Omega$ and then extracts the minimum value $h_\pi(S) = \min(\pi(S))$. The probability that the minhash function for a random permutation of a set computes the same value for two sets is equal to the Jaccard similarity of the two sets [Rajaraman and Ullman, 2010]. In practice multiple (n) hash functions are employed to reduce the variance, resulting in a vector $v^S = \langle h_{\pi_0}(S), \cdots, h_{\pi_{n-1}}(S)\rangle$ of n minimum values, one per hash function.

There are three methods we need to design for homomorphic minhash: `union`, `difference` and `score` (to compute the Jaccard estimate). However, unlike simhash for cosine similarity, the operations underlying the statistics for minhash are non-linear due to the elementwise minimum operation that produces the underlying vector of hash values. Moreover, the set semantics on which the minhash `score` method relies are problematic because the `union`

and `difference` methods need to maintain the invariance that a parent set is equal to the union of a collection of child sets: when we perform a difference operation between a parent set and a child set we cannot simply remove all of the child's elements from the parent since a sibling might also (redundantly) contribute some of those elements to the parent. We sketch a solution to these problems and include details in Appendix E. However, we note that our solution is not complete because there are several corner cases we do not yet handle. Nevertheless, we find that the representation works well in practice.

First, we handle the problem of set-semantics by augmenting the n-dimensional minhash representation v^S of each set S with an n-dimensional vector of counts c^S, in which each dimension corresponds to a minimum hash value (essentially embracing multiset semantics, but in the hash space). The count indicates the number of child sets that contribute the associated hash value. For the union of two sets $S = S_1 \cup S_2$, we either keep the count associated with the smaller hash value if they are different (that is, since we set $v^S = v^{S^\star}$ we set $c_i^S = c_i^{S^\star}$ where $S^\star = \text{argmin}_{S_j \in \{S_1, S_2\}}(v_i^{S_j})$), or sum the counts if they are the same (that is, $c_i^S = c_i^{S_1} + c_i^{S_2}$). For difference we employ the same strategy except we subtract the counts instead of add them. The counts are appropriately ignored when estimating the Jaccard since we want the estimate to reflect sets rather than multisets.

Second, we must also address the fact that the minimum is a non-linear operator. The problem arises when the count associated with a hash value becomes zero. Then, how do we recompute what the new minimum value should be? Recomputing the minimum from scratch is a nonstarter because it would require keeping around the original sparse vector representations that we are trying to supplant. Rewriting the difference in terms of unions is also computationally expensive since it would require traversing the entire tree to check what the new minimum value should be. Instead, noting that minhash typically has hundreds of hash functions (e.g., $n = 256$) for each set, we propose to ignore the hash values associated with zero counts, and employ the remaining hashes to compute the Jaccard. This strategy has consequences for both bias and variance. First, since fewer hashes are employed, the variance increases, and if left unchecked may culminate in a worst case in which all counts are zero and Jaccard can no longer be estimated. However, we can periodically refresh the counts by traversing the trees and hopefully we do not have to do such refresh operations too often in practice. Second, since the hashes associated with zero counts are correlated, the estimate is no longer unbiased. Therefore, as described in Appendix E, we modify the Jaccard estimate to make better use of the zero-counts rather than simply ignore them, and this eliminates the bias in some cases. However, we do not have a solution that works in every case.

Finally, there is also the question of how to perform union and difference for cases in which the count is zero. For now, we ignore the zero counts during these operations, but are currently exploring strategies for incorporating them to further reduce bias and variance.

3.4 Additional Compression Schemes

Hierarchical coref involves other features, such as entropy and complexity-based penalties on the bag of words representations of context and topics associated with the entities. Although not a focus of this current study, we note that some of these representations depend on the ratios of p-norms that can be estimated with Johnson-Lindenstrauss style representations.

Moreover, there exist hashing schemes to enable fast estimation of entropy. We save the homomorphic versions of these hashes for future work.

4. Experiments

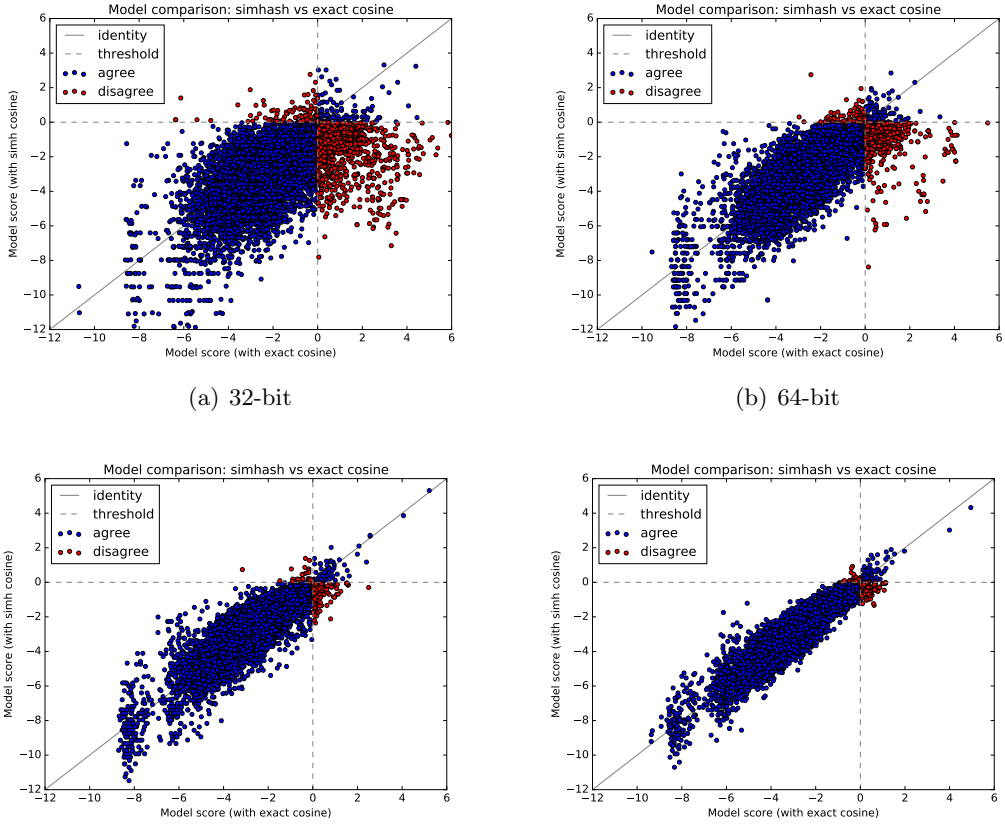
In this section we empirically study the two cosine-preserving homomorphic compression schemes, simhash and its related random projection, on a conditional random field (CRF) model of author coreference resolution, the problem introduced in Section 2. First we study simhash and we hypothesize that the this representation will substantially improve the speed of inference for the reasons outlined earlier, but it is less clear how the simhash representation will affect the quality of the model. On the one hand, our theoretical analysis shows that the variance and bias reduce rapidly with the number of bits, but on the other hand, the sheer number of vector comparisons that take place during inference is substantial, making it likely for errors to occur anyway. With this in mind, we study how simhash affects the model quality in our first set of experiments. In our second set of experiments, we study the extent to which it improves inference speed. In a third set of experiments, we compare simhash with the random projection method. Finally, we present initial results for homomorphic minhash to empirically assess how it compares to exact Jaccard, which is important given the increased bias and variance of our method.

Data We employ the REXA author coreference dataset,[2] which comprises 1404 author mentions of 289 authors with ambiguous first-initial last-name combinations: *D. Allen*, *A. Blum*, *S. Jones*, *H Robinson*, *S. Young*, *L. Lee*, *J. McGuire*, *A. Moore*. We split the data such that training set contains mentions of the first five ambiguous names (950 mentions) while the testing set comprises the remaining three ambiguous names (454 mentions). The dataset is highly ambiguous since there are multiple entities with each name. In addition, we also employ the DBLP dataset which contains over one million paper citations from which we extract about five million unlabeled author mentions [Ley, 2002].

Model We investigate homomorphic simhash in the context of the hierarchical coreference model presented in Section 2. We employ two types of feature variables, a "name bag" that represents the features of the author's name and a "context bag" that represents the remaining features in the citation from which the author mention is extracted (co-authors, title, venue, topics, keywords). For more details about the features please see Appendix B.

We employ the implementation of hierarchical coreference available in the FACTORIE toolkit, using FACTORIE's implementation of the variables, the model and the inference algorithm [McCallum et al., 2008]. We additionally implement the simhash variables and potential functions inside this framework. We employ FACTORIE's default inference algorithm for hierarchical coreference which is essentially a greedy variant of multi-try Metropolis-Hastings in which the proposals make modifications to the sub-trees (e.g., move a subtree from one entity to another, or merge two trees under a common root node). More details are in previous work, and the implementation is publicly available in FACTORIE [McCallum et al., 2009, Wick et al., 2012]. We estimate the parameters with hyper-parameter search on the training-set (Appendix B).

2. http://www.iesl.cs.umass.edu/datasets.html



(a) 32-bit

(b) 64-bit

(c) 128-bit

(d) 256-bit

Figure 1: Model score comparison with homomorphic simhash and exact sparse-vector representations. The closer points are to the identity reference line, the better. The dotted horizontal and vertical lines represent the decision threshold for inference. Points for which the two models agree are in blue, the agreements rates are: 88.6, 83.6, 97.0, 97.8 percent (respectively 32, 64, 128, 256 bits).

Experiment 1: Simhash Estimation Quality Here we study the quality of models with simhash representations by comparing them to models with exact representations. First, we compare how these models evaluate and score the intermediate results of MCMC inference. We run MCMC for 100,000 steps to optimize simhash-based models on the REXA test set (with either 256, 128, 64 and 32 bit hashes). The chains begin with the singleton configuration (all mentions in their own entity tree of size one), and at each step proposes changes to the current state which the model decides to either accept or reject. This process gradually produces larger and larger trees. For each proposed state change (sample), we record the log model ratio of both the simhash model and the exact model. We present every 10th sample in a scatter plot in Figure 1. The closer the points are to the identity reference line $y = x$, the more accurate the simhash model is for those points. Varying the number of bits has a pronounced effect on the model's ability to judge MCMC states.

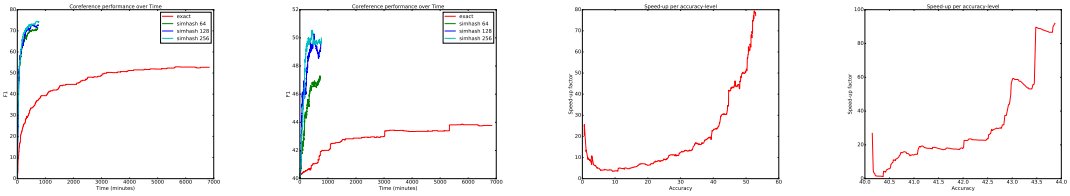
For each step, we also record if the simhash model and exact model agree upon whether to accept the stochastic proposal (blue points) or not (red points).[3] The agreement rates are 97.8, 97.0, 93.6, 88.6 percent (respectively 256, 128, 64, 32 bits). We also plot the decision boundaries for this agreement as dotted lines. The upper-left and lower-right quadrants contain all the points for which the two models disagree, while the other two quadrants contain points for which they agree. In particular, the upper-right quadrants contain the points that both the simhash model and exact model believes should be accepted (true positives), while the lower-right quadrants contain the points that both models think should be rejected (true negatives). Most points lie in this quadrant since the chance of proposing a fruitful move is relatively low. Visually, there appears to be a qualitative gap between 64 bits and 128 bits on this data, leading to a recommendation of using at least 128 bits.

We also compare the models in terms of their final coreference performance (according to B-cubed F1 [Bagga and Baldwin, 1998]). The exact model achieves an F1 of 78.6, while the simhash variants achieve F1 scores of 77.6, 75.6, 62.8, 55.6 for 256, 128, 64, 32 bits respectively. For more detailed results, with precision, recall and other types of F1, see Table 1 in the appendix. Overall, the accuracy of the 128 and 256-bit models are reasonable with 256 being competitive with the performance of the exact model. When using fewer bits, again, the performance decreases precipitously.

Experiment 2: Simhash Speed In this experiment we study the extent to which the compressed simhash representation improves inference speed. As described before, we tune the models on the REXA training set. Then, to test the models on a larger dataset, we supplement the 454 labeled REXA test mentions with five million unlabeled mentions from DBLP. We run each model on this combined dataset, initializing to the singleton configuration and then running one billion samples. We record coreference quality on the labeled subset every 10,000 samples and plot how it changes over time in Figures 2(a),2(b).

Although Experiment 1 shows that the simhash models are slightly worse in terms of their final F1 accuracy on the REXA test set, we see a striking computational advantage for simhash. For each F1 value, we compute how much faster the simhash model achieves that value than the exact model and plot this speed-improvement as a function of F1-level in Figures 2(c),2(d). As we can see, the speed improvement is more than a base-ten order of magnitude for most accuracy levels, and the speed difference increases with accuracy. This

3. These decisions, and hence agreement upon them, are deterministic with our 0-temperature sampler.



(a) MUC F1 vs time (b) B3 F1 vs time (c) Simhash speedup (MUC) (d) Simhash speedup (B3)

Figure 2: Comparison of hierarchical coreference models with either simhash or exact sparse-vector representations of the features. Simhash representations result in large speedups and have little affect on accuracy.

is because the exact representation slows down over time as the number of features in the representation grows during inference. Indeed if we look at the sampling rate over time for each model, we find that the simhash models run at about 20,000-25,000 samples per second the entire time, while the model with the exact representation starts at a rate of 3000-4000 samples per second, but then drops to under 1000 samples per second as the size of the entities get larger. This raises the question: can we improve the speed of the exact model by reducing the number of features? We address this question in Appendix C.2, but summarize here briefly: it is possible to improve the speed and reduce the gap, but simhash is still faster and there is a trade-off with coreference quality. In addition, whether feature ablation meaningfully improves performance depends on particular aspects of the data set, whereas the simhash representation can be applied generically.

Experiment 3: JL Random Projections In this experiment, we compare simhash with an approach that directly computes cosine on the statistics that underly the bit representation. As argued previously in Section 3.2, this approach is justified because the statistics are random projections that are homomorphic and cosine-preserving. Intuitively, we expect this approach to work even better because simhash further compresses these statistics into single bits, whereas in the random projection approach, we compute cosine directly on the real-valued statistics. However, our current theoretical analysis does not allow us to compare one to another; therefore, we turn to the empirical studies in this section.

We perform an experiment analogous to Experiment 1, except we employ the exact model to make decisions about what MCMC proposals to accept. This allows us to compare both approximation schemes on the same set of samples. For each accepted sample, we again ask the question how the approximate methods, simhash and JL, would have judged them. We find that JL does indeed perform better than simhash for the same number of bits.[4] In particuar, the Spearman's rho for JL increases from 93.2 to 97.2 over simhash when employing 256 bits (dimensions). For all but one case (128 bits), we find a similar improvement. Moreover, the coreference accuracy also increases in each case; for example,

4. Dimension might be a better term for JL since the floats are never converted into bits.

from 77.6 B3 F1 to 78.3. More detailed results are in Table 3 in Appendix D, along with the associated scatter plots (Figure 6).

5. Experiment 4: MinHash

We also investigate our homomorphic MinHash representation for Jaccard similarity. For lack of space, and because Jaccard is not employed by the hierarchical coreference, we relegate most of the evaluation to Appendix E.2. To briefly summarize, we find that 128 and 256 hash functions are sufficient for reasonable estimates and that the representation rarely needs to be refreshed to ensure that all the counts are above zero, enabling the computational efficiency we desire. However, there is room for improvement because only 16% of the hash functions on average have a non-zero entry after 100,000 samples. Thus, after 100,000 samples, a 256 hash version will begin to behave like a 40 hash version.

6. Related Work

Homomorphic representations have many possible uses in machine learning; for example, *homomorphic encryption* allows models to run directly on *encrypted* data for cases in which privacy is a concern [Graepel et al., 2012, Dowlin et al., 206]. Instead, our method is similar to *homomorphic compression* [McGregor, 2013] which allows our model to run directly on *compressed* data for cases in which computational efficiency is a concern. Our approach is based on simhash, a locality-sensitive hash function. Locality sensitive hashes such as simhash and `minhash` are useful in large scale streaming settings; for example, to detect duplicate webpages for web-crawlers or in nearest neighbor search [Broder, 1997b, Manku et al., 2007, Leskovec et al., 2014]. They are sometimes employed in search and machine-learning applications including coreference for "blocking" the data in order to reduce the search space [Getoor and Machanavajjhala, 2012]. Note that this application of locality sensitive hash functions for coreference is complementary to ours, and does not require it to be homomorphic. Other hashing and sketching algorithms are also employed as a strategy to compress the sufficient statistics in Bayesian models so that they can scale better as the number of parameters increase with the dataset. For example, feature hashing, count-min sketches and approximate counters have been employed in scaling latent Dirichlet categorical models such as LDA by compressing, for example, the counts that assign latent variables to mixture components [Zaheer et al., 2016, Tassarotti et al., 2018].

Our paper focuses on three homomorphic compression schemes including one for minhash. Our solution of furnishing the minhash values with counts resembles a strategy that similarly employs counts in a k minimum values (KMV) sketch for estimating the number of distinct values in a set [Beyer et al., 2007, 2009]. A key difference is that that work develops an unbiased estimator for the number of *distinct values* that explicitly involves the counts as part of the estimate itself, whereas we develop a biased estimator that directly estimates *Jaccard similarity* and that employs the counts to bound our knowledge about possible minimum hash values when they vanish during difference operations.

Our schemes are related to random projections, which are commonly used to improve computational efficiency in machine learning. Examples include feature-hashing [Weinberger et al., 2009], fast matrix decomposition [Halko et al., 2009] and fast kernel computations

[Rahimi and Recht, 2007]. However, while some of these random projections happen to be homomorphic, this property is often not exploited. The reason is that they are typically used to compress static objects that remain fixed, such as the Gram matrix for kernel methods. In contrast, our setting requires compressing dynamic sets that change during inference.

Word embeddings [Mikolov et al., 2013] also provide low-dimensional dense representations that are useful for capturing context, but in practice, these embeddings are too smooth to be useful as the sole representation for disambiguating the names of peoples, places or organizations, as is necessary in coreference (e.g., the names "Brady" and "Belichick" would be highly similar according to a word-to-vec style embedding, even though they are unlikely to be coreferent). Deep learning might allow for more suitable representations to be learnt and its application to coreference is promising. However, the current research focus is on within-document noun-phrase coreference and entity linking, while emphasizing improvements in accuracy rather than inference speed and scalability [Globerson et al., 2016, Wiseman et al., 2016, Lee et al., 2017, Raiman and Raiman, 2018]. Combining deep learning and conditional random fields for hierarchical coreference remains an open problem.

Finally, in addition to the work mentioned throughout the paper, there is an abundance of related work on coreference resolution including noun-phrase coreference, cross-document coreference, record linking, entity linking and author coreference. While not possible to cover the breadth of the space here, we refer the reader to a few useful surveys and tutorials on the subject [Ng, 2010, Getoor and Machanavajjhala, 2012]. More recently, as mentioned in the foregoing paragraph, deep learning approaches are beginning to show promise. There has also been promising recent work on scalable hierarchical clustering known as PERCH [Kobren et al., 2017]. Since PERCH employs Euclidean distance rather than cosine similarity, it currently falls outside the purview of our study. However, the Johnson-Lindenstrauss applies to Euclidean distance making random projections a possibility for PERCH.

7. Conclusion

In this paper we presented several homomorphic compression schemes for representing the sparse, high dimensional features in a graphical model for coreference resolution, even as these features change during inference. Our primary concern was cosine similarity for which we investigated simhash and angle-preserving random projections. We also proposed a homomorphic version of minhash for Jaccard similarity. In the particular case of simhash, we presented a new variant of the original estimator and analyzed its statistical properties — including variance, bias and consistency — to help justify its use in a probabilistic model. We found that both simhash and angle-preserving random projections were sufficiently accurate, given enough dimensions, to represent features for coreference, and that the random projections produce slightly better estimates. Moreover, these representations were an order of magnitude faster than conventional sparse data structures, laying the foundation for greater scalability.

References

Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66:671–687, 2003.

Amit Bagga and Breck Baldwin. Algorithms for scoring coreference chains. In *Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, 1998.

Kevin Beyer, Peter J. Haas, Berthold Reinwald, Yannis Sismanis, and Rainer Gemulla. On synopses for distinct-value estimation under multiset operations. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 199–210, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-686-8.

Kevin Beyer, Rainer Gemulla, Peter J. Haas, Berthold Reinwald, and Yannis Sismanis. Distinct-value synopses for multiset operations. *Commun. ACM*, 52(10):87–95, October 2009. ISSN 0001-0782.

Andrei Z Broder. On the resemblance and containment of documents. In *Compression and complexity of sequences 1997. proceedings*, pages 21–29. IEEE, 1997a.

Andrei Z. Broder. On the resemblance and containment of documents. In *IEEE SEQUENCES*, 1997b.

Andrei Z Broder, Moses Charikar, Alan M Frieze, and Michael Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.

Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, 2002.

Aron Culotta, Pallika Kanani, Robert Hall, Michael Wick, and Andrew McCallum. Author disambiguation using error-driven machine learning with a ranking loss function. In *Sixth International Workshop on Information Integration on the Web (IIWeb-07), Vancouver, Canada*, 2007a.

Aron Culotta, Michael L. Wick, and Andrew McCallum. First-order probabilistic models for coreference resolution. In *HLT-NAACL*, 2007b.

Nathan Dowlin, ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *ICML*, 206.

Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009. ISBN 978-0-521-88427-3.

Lise Getoor and Ashwin Machanavajjhala. Entity resolution: Tutorial. In *VLDB Tutorial*, 2012.

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. Collective entity resolution with multi-focal attention. In *ACL*, 2016.

Thore Graepel, Kristin Lauter, and Michael Naehrig. Ml confidential: Machine learning on encrypted data, 2012.

N. Halko, P.G. Martinsson, and J.A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev*, 53(2), 2009.

Hui Han, Lee Giles, Hongyuan Zha, Cheng Li, and Kostas Tsioutsiouliklis. Two supervised learning approaches for name disambiguation in author citations. In *Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '04, pages 296–305, New York, NY, USA, 2004. ACM.

Monika Rauch Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*, pages 284–291, 2006. doi: 10.1145/1148170.1148222. URL http://doi.acm.org/10.1145/1148170.1148222.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.

William B. Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. In *Conference in modern analysis and probability*, pages 189–206. Amer. Math. Soc., 1982.

Ari Kobren, Nicholas Monath, Akshay Krishnamurthy, and Andrew McCallum. A hierarchical algorithm for extreme clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 255–264, New York, NY, USA, 2017. ACM.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. End-to-end neural coreference resolution. In *EMNLP*, 2017.

J. Leskovec, A. Rajaraman, and J.D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2014. ISBN 9781107077232.

Michael Ley. The DBLP computer science bibliography: Evolution, research issues, perspectives. In *String Processing and Information Retrieval, 9th International Symposium, SPIRE 2002, Lisbon, Portugal, September 11-13, 2002, Proceedings*, pages 1–10, 2002.

Avner Magen. Dimensionality reductions that preserve volumes and distance to affine spaces, and their algorithmic applications. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 239–253. Springer, 2002.

Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. Detecting near-duplicates for web crawling. In *WWW*, 2007.

Andrew McCallum and Ben Wellner. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proceedings of the 2003 International*

Conference on Information Integration on the Web, IIWEB'03, pages 79–84. AAAI Press, 2003.

Andrew McCallum, Kashayar Rohanemanesh, Michael Wick, Karl Schultz, and Sameer Singh. FACTORIE: efficient probabilistic programming for relational factor graphs via imperative declarations of structure, inference and learning. In *NIPS workshop on Probabilistic Programming*, 2008.

Andrew McCallum, Karl Schultz, and Sameer Singh. FACTORIE: Probabilistic programming via imperatively defined factor graphs. In *Neural Information Processing Systems (NIPS)*, 2009.

Andrew McGregor. Towards a thoery of homorphic compression. In P. Bonizzoni, V. Brattka, and B. Lowe, editors, *The nature of computational. Logic, algorithms, applications.*, pages 316–319. Springer, 2013.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

Nicolas Monath and Andrew McCallum. University of massachusett's entry in the uspto patentsview workshop. In *PatentsView Inventor Disambiguation Workshop*, 2016.

H.B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic linkage of vital records. *Science*, 130:954–959, 1959.

Vincent Ng. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1396–1411, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 492–501, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Ali Rahimi and Ben Recht. Random features for large-scale kernel machines. In *NIPS*, 2007.

Jonathan Raiman and Olivier Raiman. Deeptype: Multilingual entity linking by neural type system evolution. In *AAAI*, 2018.

A Rajaraman and JD Ullman. Finding similar items. *Mining of Massive Datasets*, 77:73–80, 2010.

Joseph Tassarotti, Jean-Baptiste Tristan, and Michael Wick. Sketching for latent dirichlet-categorical models, 2018.

PatentsView Inventor Disambiguation Workshop, 2016. United States Patent and Trademark Office (USPTO). URL https://www.uspto.gov/about-us/organizational-offices/office-policy-and-international-affairs/patentsview-inventor.

A.W. Van Der Vaart. *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics, 3. Cambridge University Press, 1998. ISBN 9780521496032. URL https://books.google.com/books?id=udhfQgAACAAJ.

Kilian Weinberger, Anibarn Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *ICML*, 2009.

Michael Wick, Sameer Singh, and Andrew McCallum. A discriminative hierarchical model for fast coreference at scale. In *ACL*, 2012.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. Learning global features for coreference resolution. In *NAACL-HLT*, 2016.

Manzil Zaheer, Michael L. Wick, Jean-Baptiste Tristan, Alex Smola, and Guy L. Steele Jr. Exponential stochastic cellular automata for massively parallel inference. In *ICML*, 2016.

Appendix A. Properties of SimHash

Let a and b be vectors on the unit hypersphere in the Euclidean space $\mathcal{R}^d$ with angle θ between them, and let u be a random d-dimensional vector whose coordinates are sampled independently from $N(0,1)$. Note that the probability that the two vectors end up on a different side of the hyperplane is

$$\mathbb{P}(sign(a \cdot u) \neq sign(b \cdot u)) = 1 - \mathbb{P}(sign(a \cdot u) = sign(b \cdot u)) \tag{10}$$

$$= 1 - \frac{\theta}{\pi} \tag{11}$$

Let the random variable X have value 1 if a and b are on different sides of u, and 0 otherwise. Then:

$$\mathbb{E}[X] = 1 - \frac{\theta}{\pi} \tag{12}$$

$$\mathbb{V}[X] = \frac{\theta}{\pi}\left(1 - \frac{\theta}{\pi}\right) \tag{13}$$

If we let the family of random variables $X_1, ..., X_n$ be the result of repeating this process several times with independently randomly chosen hyperplanes, then we have:

$$\mathbb{E}[\frac{1}{n}\sum_{i=1}^{n} X_i] = 1 - \frac{\theta}{\pi} \tag{14}$$

$$\mathbb{V}[\frac{1}{n}\sum_{i=1}^{n} X_i] = \frac{1}{n^2}\sum_{i=1}^{n}\mathbb{V}[X_i] = \frac{1}{n}\cdot\frac{\theta}{\pi}\left(1 - \frac{\theta}{\pi}\right) \tag{15}$$

Set $\overline{X}_n = \frac{1}{n}\sum_{i=1} X_i$. Then we have that $\mathbb{E}[\pi(1 - \overline{X}_n)] = \theta$, suggesting that we use $\cos(\pi(1 - \overline{X}_n))$ as an estimator for $\cos(\theta)$. Let $g(x) = \cos(\pi(1 - x))$ and $C_n = g(\overline{X}_n)$ be this estimator. We now explore some properties of C_n.

Lemma A.1. *C_n is consistent. In particular, $C_n \xrightarrow{a.s.} \cos(\theta)$.*

Proof. By the strong law of large numbers, we have that $\overline{X}_n \xrightarrow{\text{a.s.}} 1 - \frac{\theta}{\pi}$. Since g is continuous, by the continuous mapping theorem [Van Der Vaart, 1998],

$$g(\overline{X}_n) \xrightarrow{\text{a.s.}} g(1 - \frac{\theta}{\pi}) = \cos(\theta)$$

□

Lemma A.2. *$\mathbb{E}[C_n] = \cos(\theta) + E_n$, where $|E_n| \leq \frac{\pi^2}{8n}$*

Proof. Set $\mu = \mathbb{E}[\overline{X}_n] = 1 - \frac{\theta}{\pi}$. The first degree Taylor series for $g(x)$ about μ is:

$$g(x) = \cos(\theta) + \pi\sin(\theta)(x - \mu) + R(x)$$

where R is the remainder term. Applying Lagrange's remainder formula, we have that for each x, there exists some $h(x)$ lying between μ and x such that:

$$R(x) = \frac{\pi^2\cos(\pi h(x))}{2}(x - \mu)^2$$

We have then that:

$$\mathbb{E}[g(\overline{X}_n)] = \cos(\theta) + \pi\sin(\theta)(\mathbb{E}[\overline{X}_n] - \mu) + \mathbb{E}[R(\overline{X}_n)] \tag{16}$$
$$= \cos(\theta) + \mathbb{E}[R(\overline{X}_n)] \tag{17}$$

Thus it suffices to bound $|\mathbb{E}[R(\overline{X}_n)]|$. We have:

$$|\mathbb{E}[R(\overline{X}_n)]| \leq \mathbb{E}[|R(\overline{X}_n)|] \tag{18}$$
$$= \mathbb{E}[|\frac{\pi^2\cos(\pi h(\overline{X}_n))}{2}|(\overline{X}_n - \mu)^2] \tag{19}$$
$$\leq \mathbb{E}[\frac{\pi^2}{2}(\overline{X}_n - \mu)^2] \tag{20}$$
$$= \frac{\pi^2}{2}\mathbb{V}[\overline{X}_n] \tag{21}$$
$$= \frac{\pi^2}{2}\cdot\frac{1}{n}\cdot\frac{\theta}{\pi}(1-\frac{\theta}{\pi}) \tag{22}$$
$$\leq \frac{\pi^2}{2}\cdot\frac{1}{4n} = \frac{\pi^2}{8n} \tag{23}$$

□

Lemma A.3. $\mathbb{V}[C_n] = \frac{\pi^2\sin^2(\theta)}{n}\cdot\frac{\theta}{\pi}(1-\frac{\theta}{\pi}) + O(n^{-3/2})$

Proof. Before delving into the details, let us sketch the idea. The Taylor approximation for g shows us that $g(x) \approx \cos(\theta) + \pi\sin(\theta)(x-\mu)$. So, recalling that $\mathbb{V}[C_n] = \mathbb{V}[g(\overline{X}_n)] = \mathbb{E}[g(\overline{X}_n)^2] - \mathbb{E}[g(\overline{X}_n)]^2$, and plugging in the approximation we get:

$$\mathbb{V}[g(\overline{X}_n)] \approx \mathbb{E}[\big(\cos(\theta) + \pi\sin(\theta)(\overline{X}_n - \mu)\big)^2] - \cos(\theta)^2 \tag{24}$$
$$= 2\pi\sin(\theta)\mathbb{E}[\overline{X}_n - \mu] + \pi^2\sin^2(\theta)\mathbb{E}[(\overline{X}_n - \mu)^2] \tag{25}$$
$$= \pi^2\sin^2(\theta)\mathbb{E}[(\overline{X}_n - \mu)^2] \tag{26}$$

Now, to get the actual error term, we carry out the same process without dropping the remainder term $R(x)$ from the Taylor expansion. We have:

$$\mathbb{V}[g(\overline{X}_n)] = \mathbb{E}[\big(\cos(\theta) + \pi\sin(\theta)(\overline{X}_n - \mu) + R(\overline{X}_n)\big)^2] - \big(\cos(\theta) + \mathbb{E}[R(\overline{X}_n)]\big)^2 \tag{27}$$
$$= \cos(\theta)^2 + 2\cos(\theta)\mathbb{E}[R(\overline{X}_n)] + \mathbb{E}[R(\overline{X}_n)^2] + 2\pi\cos(\theta)\sin(\theta)\mathbb{E}[\overline{X}_n - \mu] \tag{28}$$
$$+ 2\pi\sin(\theta)\mathbb{E}[R(\overline{X}_n)(\overline{X}_n - \mu)] + \pi^2\sin^2(\theta)\mathbb{E}[(\overline{X}_n - \mu)^2] \tag{29}$$
$$- \big(\cos(\theta) + \mathbb{E}[R(\overline{X}_n)]\big)^2 \tag{30}$$
$$= \pi^2\sin^2(\theta)\mathbb{V}[\overline{X}_n] - \mathbb{E}[R(\overline{X}_n)]^2 + \mathbb{E}[R(\overline{X}_n)^2] + 2\pi\sin(\theta)\mathbb{E}[R(\overline{X}_n)(\overline{X}_n - \mu)] \tag{31}$$

It suffices to bound the last three terms. From the previous result, we know that the first of these three terms is the square of something which is $O(1/n)$. For the second,

$$|\mathbb{E}[R(\overline{X}_n)^2]| = \mathbb{E}[R(\overline{X}_n)^2] \tag{32}$$

$$= \mathbb{E}[\frac{\pi^4 \cos^2(\pi h(\overline{X}_n))}{4}(\overline{X}_n - \mu)^4] \tag{33}$$

$$\leq \frac{\pi^4}{4}\mathbb{E}[(\overline{X}_n - \mu)^4] \tag{34}$$

Let $p = \theta/\pi$. Then

$$\mathbb{E}[(\overline{X}_n - \mu)^4] = \frac{3np^4 - 6np^3 + 3np^2 - 6p^4 + 12p^3 - 7p^2 + p}{n^3} \tag{35}$$

$$\leq \frac{6n + 13}{n^3} \tag{36}$$

so that

$$|\mathbb{E}[R(\overline{X}_n)^2]| \leq \frac{\pi^4(6n + 13)}{4n^3} \tag{37}$$

For the final term, we have:

$$|\mathbb{E}[R(\overline{X}_n)(\overline{X}_n - \mu)]| \leq \sqrt{\mathbb{E}[R(\overline{X}_n)^2]\mathbb{E}[(\overline{X}_n - \mu)^2]} \tag{38}$$

$$\leq \sqrt{\frac{\pi^4(6n + 13)}{n^3} \cdot \frac{1}{n} \cdot \frac{\theta}{\pi}(1 - \frac{\theta}{\pi})} \tag{39}$$

$$\leq \sqrt{\frac{\pi^4(6n + 13)}{n^4} \cdot \frac{\theta}{\pi}(1 - \frac{\theta}{\pi})} \tag{40}$$

which is $O(n^{-3/2})$ □

Lemma A.4. $\mathbb{P}(|C_n - \cos(\theta)| > \delta + \frac{\pi^2}{8n}) \leq 2e^{-2n\delta^2/\pi^2}$

Proof. Define $f_n : [0, 1]^n \to \mathbb{R}$ by $f_n(x_1, \ldots, x_n) = \cos(\pi(1 - \frac{1}{n}\sum_{i=1}^n x_i))$. Then $f_n(X_1, \ldots, X_n) = C_n$. We now show that for all i, and $x_1, \ldots, x_n \in [0, 1]^n$ and $x_i' \in [0, 1]$,

$$|f_n(x_1, \ldots, x_i, \ldots, x_n) - f_n(x_1, \ldots, x_i', \ldots, x_n)| \leq \frac{\pi}{n}|x_i - x_i'|$$

Fix i and $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$, and consider the function

$$h(x) = f_n(x_1, \ldots, x_{i-1}, x, \ldots, x_n)$$

Then, by the mean value theorem, for all $x_i, x_i' \in [0, 1]$, there exists some $x^* \in [0, 1]$ such that:

$$h(x_i) - h(x_i') = h'(x^*)(x_i - x_i')$$

$$= \frac{\pi}{n}\sin\left(\pi\left(1 - \frac{1}{n}(x_1 + \cdots + x^* + \cdots + x_n)\right)\right)(x_i - x_i')$$

Hence $|h(x_i) - h(x_i')| \leq \frac{\pi}{n}|x_i - x_i'|$. Since the function f_n is Lipschitz continuous in each coordinate with parameter $\frac{\pi}{n}$, by the method of bounded differences [Dubhashi and Panconesi, 2009, Corollary 5.2], we have:

$$\mathbb{P}(|f_n(X_1, \ldots, X_n) - \mathbb{E}[f_n(X_1, \ldots, X_n)]| > \delta) \leq 2e^{-2n\delta^2/\pi^2}$$

Moreover, from above we have that $|\mathbb{E}[C_n] - \cos(\theta)| < \frac{\pi^2}{8n}$. Hence,

$$\mathbb{P}(|C_n - \cos(\theta)| > \delta + \frac{\pi^2}{8n}) \leq 2e^{-2n\delta^2/\pi^2}$$

□

The above bound is uniform over θ, but stronger bounds for very small or large values of θ can be obtained as follows: Set $p = 1 - \theta/\pi$ and apply standard Chernoff bounds to the binomial variable $\overline{X}_n$. Then observe that if $\overline{X}_n$ deviates from its mean by no more than δ with high probability, then $g(\overline{X}_n)$ is within $|\sup_{p-\delta \leq t \leq p+\delta} g'(t)|\delta$ around $\cos(\theta)$ with at least the same high probability.

Appendix B. Model and parameter estimation

Features Here we provide more details about the features. First recall that we employ two types of feature variables, a "name bag" that represents the features of the authors name and a "context bag" that represents the remaining features in the citation from which the author mention is extracted. We populate the "name" bag of each mention with the full name, the first-initial last name, and character tri-grams of the author's name string as it appears in the mention. We populate the "context" features of each mention with the title and venue, the co-authors with whom the paper is authored, author-provided keywords for the paper. For the title we employ white-space tokenization, character 3-grams and 4-grams, for venue we employ white-space tokenization as well as the entire string and for co-authors we employ the first-initial last-name of all co-authors. Finally for topics, we take the top 3 topics for each citation as determined by a 200-topic LDA model trained on all of DBLP and REXA combined. We employ the training data to experiment with several feature combinations, including using all 200-topics for each citation, but found that this combination of features works best.

For the exact sparse-vector representations, we employ the default implementation in FACTORIE, which employs hashmaps to store the non-zero features and their corresponding weights. For the simhash representation we employ a java implementation of `murmurhash3` to hash each feature to construct the initial dense bit vectors that underly the homomorphic simhash representation of the mentions. These vectors are then added and subtracted during inference to produce the underlying vectors of the entities and subentities on which the actual bits are computed.

Parameter estimation Following previous work, we manually tune the model with exact feature representations on the training data with the help of hyper-parameter search. We use identical parameters for the simhash model, except we lower the translation value for the context bag to compensate for the extra variance that can increase the chance of false-positives. More details follow below.

Model	B3			MUC		
	Precision	Recall	F1	Precision	Recalll	F1
exact	75.3	82.2	78.6	88.3	91.2	89.7
simhash 256	74.3	81.2	77.6	86.5	88.8	87.7
simhash 128	76.1	75.1	75.6	86.8	87.1	86.9
simhash 64	66.0	60.0	62.8	81.2	82.4	81.8
simhash 32	58.0	53.4	55.6	75.8	80.3	78.0
jl 258	73.5	83.9	78.3	87.1	91.2	89.1
jl 128	74.9	77.1	75.9	87.4	87.4	87.4
jl 64	77.2	60.8	68.0	85.7	82.6	84.1
jl 32	74.6	60.0	66.5	84.4	82.6	83.6

Table 1: Author coreference evaluation on REXA with B3 and MUC evaluation measure.

The model comprises only a small handful of parameters: a weight and translation parameter for each bag, and a prior on the number of entities and subentities. Since the prior is somewhat redundant to the translation parameters, we fix the prior to zero and do not tune it. We also fix the parameter for the number of subentities to -0.5 to penalize deeper trees. Then, we tune the remaining parameters manually with the assistance of hyper-parameter search. We explore weight values in the set $\{0.0, 1.0, 2.0, 4.0\}$ and translation values in the set $\{-0.75, -0.5, -0.25, 0.0\}$. For the model that employs exact feature representations, we found a weight of 4.0 and translation of -0.5 works best on the training data for the model that employs exact feature representations. Since the variance of the Simhash estimator increases with the angles, false-positives become a potential issue. Therefore, to compensate for this extra variance, we set the translation for the context bag of the simhash model more aggresively to -0.6 via binary search with the training data and 128-bit model (the other simhash models were not further tuned and employ the same parameters as the 128-bit model).

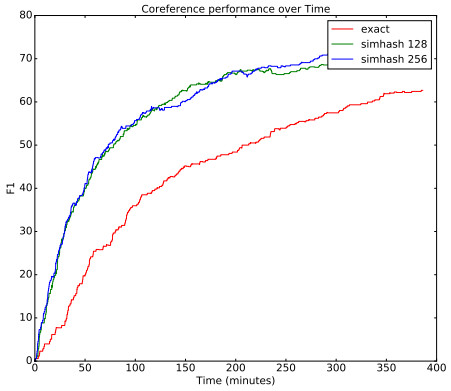
Appendix C. Detailed coreference results

C.1 Evaluation on REXA

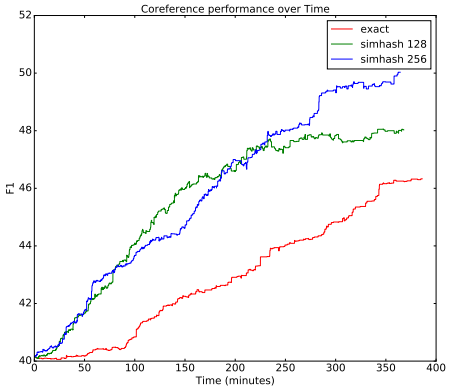
Table 1 contains a more detailed set of results on the REXA author coreference data. We evaluate with both B-cubed (B3) and MUC precision, recall and F1. The results in this table employ all the features of the author mention. For feature ablations, see Table 2 in the sequel.

C.2 Feature ablations

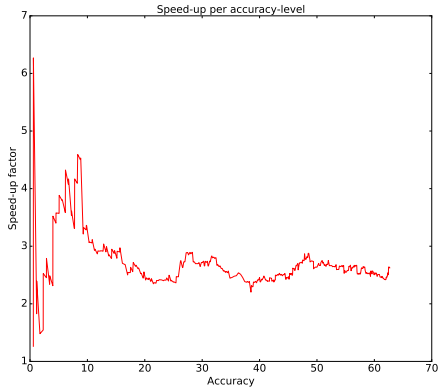
As we speculate in Section 4, we might also be able to improve the speed of the exact model by including fewer features since this would reduce the cost of dynamically updating the sparse-vector representations. To this end, we perform some feature ablations by trying various subsets of the six feature-types: name, topic, co-authors, venue, keywords, and title (with and without n-grams), re-tuning the model on the training set and then reporting the accuracy on the test-set. Note that for the exact model, changing the number of features



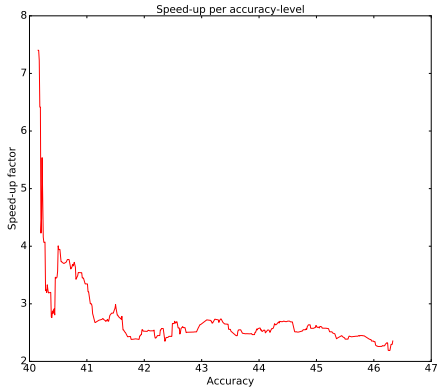
(a) MUC F1 vs time



(b) B3 F1 vs time



(c) Simhash speedup (MUC)



(d) Simhash speedup (B3)

Figure 3: Comparison of the hierarchical coreference models that employ either simhash or exact sparse-vector representations of the features. We removed the n-gram features to improve the run-time of the exact representation on the larger dataset, but this comes at the expense of lower accuracy.

Features	Sample/sec	Precision	Recall	F1 (B-cubed)
name	16750	77.0	44.6	56.5
+topics	8733	75.6	57.5	65.3
+co-authors	1269	75.7	52.3	61.9
+venue	7451	48.3	76.6	59.2
+title	1973	73.6	87.4	79.9
+topics+coauth	13262	75.9	56.9	65.0
all but title	6858	71.0	64.8	67.7
all with title (no n-grams)	10309	71.4	85.2	75.9
all	1835	72.6	86.0	78.7
all (simhash 256 bit)	16556	75.2	81.4	78.2
all (simhash 128 bit)	19841	72.6	82.3	77.2
all (simhash 64 bit)	22988	64.7	65.7	65.2
all (simhash 32 bit)	19417	57.4	59.1	58.3

Table 2: Feature ablations on the test set. B-cubed precision (p), recall (r), and F1. We can see that more features help, but come at a price: slower inference (measured as samples per second).

affects both the quality of the model and its speed, but for the simhash model, it only affects the quality since the representation is fixed in dimension. Therefore, in the spirit of the tradeoff, we include all features for simhash and instead vary the number of bits.

For the feature ablation study, we record the samples per second and the B-cubed F1 on the REXA test set and plot these points in Figure 5 with speed (samples per second) on the x-axis and F1 on the y-axis. Points in the upper-left correspond to models that are accurate, but slow, while points in the lower-right correspond to models that are fast, but inaccurate. Points in the upper-right correspond to models that are both fast and accurate. As we can see, varying the number of features for the exact model indeed improves the performance, but it often comes at the expense of accuracy. The simhash models on the other hand (indicated by red triangles) are able to achieve both speed and accuracy, given enough bits. This scatter plot is intended to understand the trade-off space and we omit the labels on the points to reduce clutter. Instead, we provide these details in a separate table (Table 2).

From these ablations we can see that the best speed-accuracy trade-off for the exact model employs all the features except for the title n-grams. Indeed, the title n-grams are the main culprit for slowing down the exact model since they explode the number of features. Thus, we repeat the experiment for Figure 2 (Experiment 2 in Section 4), with this reduced set of features and manage to improve the running time of the exact model to the point at which the simhash model is only 2-3 times faster. In these experiments, the exact model is much faster than when it had to cope with the entire set of features, and achieves around 7000 samples per second before eventually falling to 5000 as the entities get larger.

While reducing the features is a reasonable strategy for improving the inference speed without resorting to simhash, it also has several drawbacks. First, reducing the number of features can have a negative affect on the accuracy, as we see in Figure 5. When run

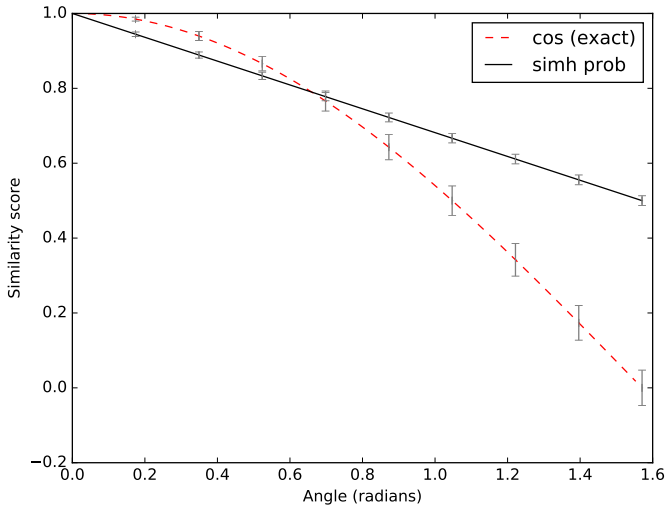


Figure 4: A comparison of exact cosine with two sim-hash approximations. The first is the usual linear approximation, depicted as the linear curve with error bars. The second is as an estimator for cos(*theta*), depicted as the error bars on the exact cosine curve (standard-error of 32-bit hash).

exclusively on the labeled data, the final F1 drops to 75.9, which is lower than the 256-bit simhash model (77.6 F1), while also being twice as slow. In addition, this strategy of reducing the feature space will not work as well in other coreference domains. For example, in author coreference, the true size of author entities is limited by the number of papers the author publishes. It is unlikely that many authors publish more than 1000 papers, and the average is likely to be much smaller. Thus, we can reasonably manage the size of the entity and sub-entity bags simply by reducing the number of features in each mention. In contrast, consider the problem of cross-document coreference on newswire data. The entity "United States" might be mentioned more than a million times in vastly different contexts. Thus, even if we limited the feature-space in the beginning, it is likely that these large entities will still accumulate a large number of features anyway. Models based on simhash will not succumb to this problem. Finally, the simhash representation is appealing because it allows a practitioner to focus on achieving high accuracy with the full flexibility and freedom when engineering features, without having to worry about the running-time performance (an experiment like the type we did in Experiment 1 would help the practitioner set the number of bits without the need for labeled data).

C.3 Note on Memory Use

Our implementation employs 32-bit floats to represent the homomorphic simhash statistics, which results in a memory use of $k * 32$, or between 1kb-8kb per node depending on the number of dimensions. This could be substantially reduced using integer representations

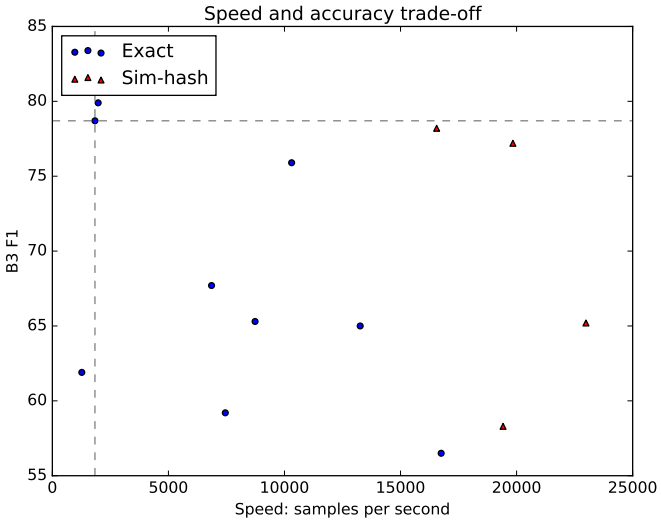


Figure 5: Accuracy versus speed as we vary the number of features for the exact model and number of bits for simhash.

with fewer bits or approximate counters. Although this would still require more memory than non-homomorphic simhash, in the context of hierarchical coreference, the memory use is still small, and often better than the original representation.

Appendix D. Johnson-Lindenstrauss Results

Here we include the details results associated with Experiment 3 in Section 4 in the main paper.

In Figure 6, we compare simhash to the random projection approach, which we abbreviate as JL (for Johnson-Lindenstrauss, since it falls under the purvue of these lemmas). The experiment is analogous to Experiment 1, except that when we run MCMC inference to obtain the data points, we employ the exact model to evaluate the MCMC moves and make acceptance decisions in order to allow for both approximation schemes to be compared on the same set of samples. For accepted sample, we can again ask the question how the approximate methods, simhash and JL, would have judged the MCMC sample. We display the results in a scatter plot with exact model judgements on the x-axis and the approximate method on the y-axis. In the figure, there are eight subplots. The left column contains the simhash comparison and the right column contains the JL comparison. The number of bits increases with each row (32, 54, 128, 256).

In Table 3 we report a numerical comparison to corroborate the scatter plots. In particular, for a fixed number of dimensions, we compare the two methods in terms of the coreference F1 (B-cubed and MUC), their correlation (Spearman's rho) with the exact model, and their accuracy with respect to exact model's decisions in accepting and rejecting

Bits	Algorithm	B3 F1	MUC f1	Spearman's rho	Acc
256	simhash	77.6	87.7	93.2	97.8
	jl	78.3	89.1	97.2	99.3
128	simhash	75.6	86.9	96.4	88.3
	jl	75.9	87.4	95.2	98.3
64	simhash	62.8	81.8	81.2	93.1
	jl	68.0	84.1	90.1	96.9
32	simhash	55.6	78.0	70.0	93.7
	jl	66.5	83.6	81.9	94.6

Table 3: A comparison of SimHash and JL style cosine computations. Spearman's rho and accuracy are computed with respect to the exact cosine computations.

MCMC proposals. We find that a model based on JL cosine estimates are better correlated with the exact model, and this translates into small improvements in coreference F1.

Appendix E. MinHash

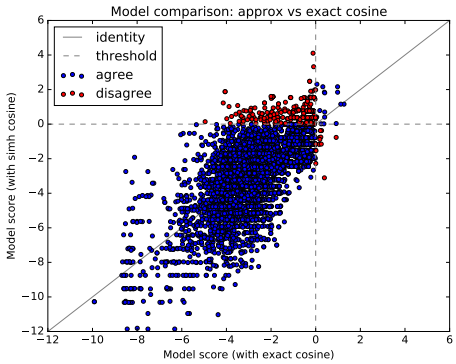
E.1 Homomorphic MinHash

Background: MinHash Minhash [Broder, 1997a, Broder et al., 2000] is a locality-sensitive hash function for Jaccard similarity, defined for binary vectors. Note that binary vectors can also represent sets such that, the non-zero entries correspond to elements in the set. For example, a binary vector in $\mathbb{R}^d$ is equivalent to a set $\Omega = \{0, 1, 2, \ldots, d-1\}$. The Jaccard similarity is a measure of resemblance between two sets $S_1, S_2 \in \Omega$, defined as $J = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$. Minhash applies a random permutation (which can be accomplished using a random hash function e.g., seeded 64-bit murmur hash) $\pi : \Omega \to \Omega$, on a given set $S \subset \Omega$ and then extracts the minimum value $h_\pi(S) = \min(\pi(S))$. The probability that the minhash function for a random permutation of a set computes the same value for two sets is equal to the Jaccard similarity of the two sets ([Rajaraman and Ullman, 2010] illustrates an intuitive derivation). Formally,

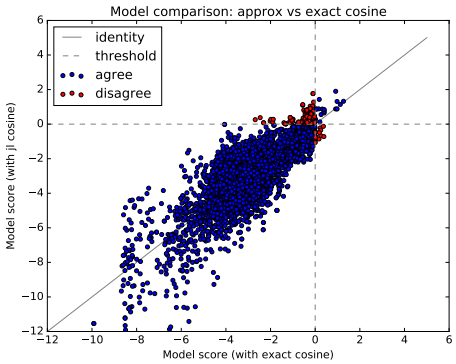
$$Pr(h_\pi(S_1) = h_\pi(S_2)) = \frac{\mid S_1 \cap S_2 \mid}{\mid S_1 \cup S_2 \mid} = J \tag{41}$$

The Jaccard estimate is then given by $\hat{J} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\left[h_{\pi_i}(S_1) = h_{\pi_i}(S_2)\right]$. The variance of this estimate is $\text{Var}(\hat{J}) = \frac{1}{n} J(1-J)$.

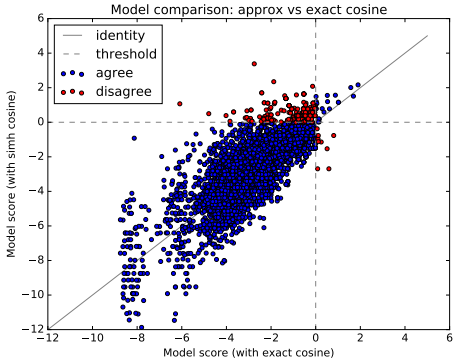
Homomorphic MinHash: Unlike simhash for cosine similarity, the operations underlying the statistics for minhash are non-linear due to the element-wise minimum operation that produces the underlying vector of hash values. Moreover, the set semantics on which minhash relies are problematic because we need to maintain the invariance that a parent set is equal to the union of a collection of child sets: when we perform a difference operation between a parent set and a child set we cannot simply remove all of the child's elements from the parent since a sibling might also (redundantly) contribute some of those elements to the parent.



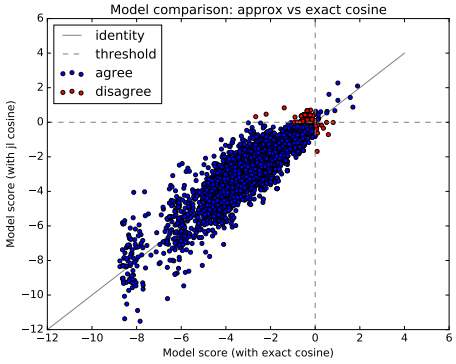
(a) 32-bit



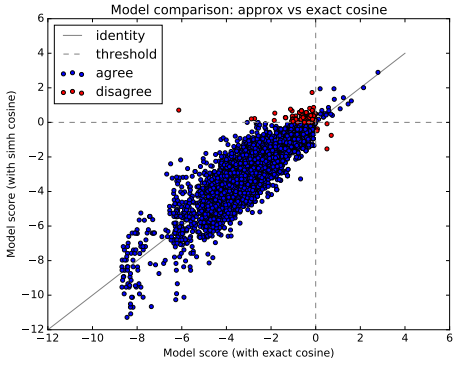
(b) 32-dim



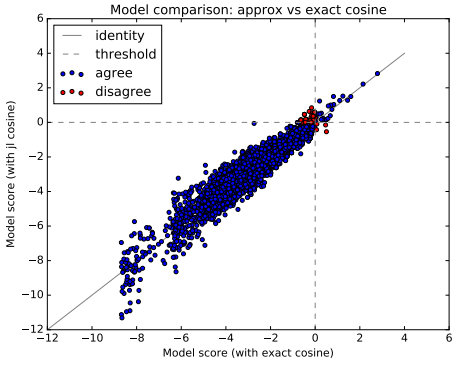
(c) 64-bit



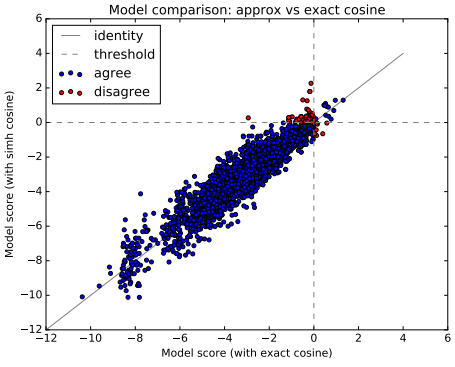
(d) 64-dim



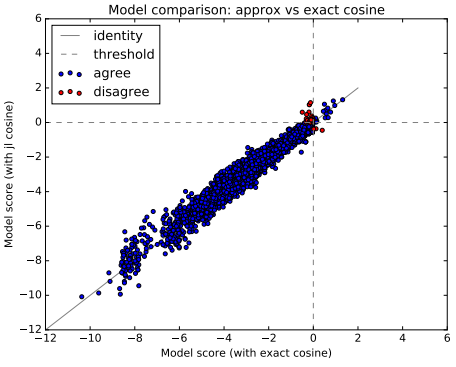
(e) 128-bit



(f) 128-dim



(g) 256-bit



(h) 256-dim

To address this situation, we introduce additional non-negative integer variables as counts that indicate the number of child sets that contribute the associated hash value. We augment the n-dimensional minhash representation v^S of each set S with an n-dimensional vector of counts c^S, in which each dimension corresponds to a minimum hash value. This representation is similar to multiset semantics, but is in the hash space instead.

We observe that given the minhash values of two sets, it is relatively straightforward to compute the minhash value of the union which is simply the minimum of the minhash values of the two nodes being merged, i.e., $h_{\pi_i}(S_1 \cup S_2) = \min(h_{\pi_i}(S_1), h_{\pi_i}(S_2))$. To obtain the corresponding counts, we either keep the count associated with the smaller hash value if they are different (that is, we set $c_i^{S_1 \cup S_2} = c_i^{S^\star}$ where $S^\star = \text{argmin}_{S_j \in \{S_1, S_2\}}(h_{\pi_i}(S_j))$, or sum the counts if they are the same (that is, $c_i^{S_1 \cup S_2} = c_i^{S_1} + c_i^{S_2}$).

In case of set difference however, the computation is not trivial (and as we mentioned earlier, the operation is not the standard set difference either). We consider the following possible cases to illustrate our approach.

(a) $h_{\pi_i}(S_1) < h_{\pi_i}(S_2)$ implies that in the original feature space, there is an element e such that $e \in S_1$ and $e \notin S_2$. In this case, $h_{\pi_i}(S_1 \setminus S_2) = h_{\pi_i}(S_1)$ and the corresponding count is equal to $c_i^{S_1}$.

(b) $h_{\pi_i}(S_1) > h_{\pi_i}(S_2)$ implies that in the original feature space, there is an element e such that $e \in S_2$ and $e \notin S_1$. Again in this case, $h_{\pi_i}(S_1 \setminus S_2) = h_{\pi_i}(S_1)$ and the corresponding count is equal to $c_i^{S_1}$. Note that since we assume that a node being subtracted was previously added to current node at some point, we can eliminate this case.

(c) $h_{\pi_i}(S_1) = h_{\pi_i}(S_2)$ i.e., the minhash values are equal. In this case, if $c_i^{S_1} > c_i^{S_2}$, $h_{\pi_i}(S_1 \setminus S_2) = h_{\pi_i}(S_1)$ and the count is $c_i^{S_1} - c_i^{S_2}$. We do not consider the case $c_i^{S_1} < c_i^{S_2}$ because of the same assumption as above, i.e., a node being subtracted was previously added to current node.

However, when $c_i^{S_1} = c_i^{S_2}$, removing S_2 from S_1 leads to loss of information that can not be retrieved without referring back to the original feature space. Recomputing the minimum from scratch is a nonstarter because it would require keeping around the original sparse vector representations that we are trying to supplant. Rewriting the difference in terms of unions is also computationally expensive since it would require traversing the entire tree to check what the new minimum value should be. Instead, noting that minhash typically has hundreds of hash functions (e.g., $n = 256$) for each set, we propose to ignore the hash values associated with zero counts, and employ the remaining hashes to compute the Jaccard.

This strategy of ignoring hash values that have zero counts has consequences for both bias and variance. First, since fewer hashes are employed, the variance increases, and if left unchecked may culminate in a worst case in which all counts are zero and Jaccard can no longer be estimated. However, we can periodically refresh the counts by traversing the trees and hopefully we do not have to do such refresh operations too often in practice.

Second, since the hashes associated with zero counts are correlated, the estimate is no longer unbiased. For example, consider the case when we merge the node $S_1 = \{\text{mustard}\}$

with the node $S_2 = \{\text{hot-dog, mustard}\}$. Now, if we remove S_2 from $S_1 \cup S_2$, the hash values corresponding to the element "mustard" become zero. Consequently, the differentiating information between $(S_1 \cup S_2) \setminus S_2$ and S_2 is lost resulting in a biased Jaccard estimate, i.e., $\hat{J}((S_1 \cup S_2) \setminus S_2, S_2) = 1$. To address this problem, we modify the Jaccard estimate to make better use of the zero-counts rather than simply ignore them, and this eliminates the bias in some cases. However, we do not have a solution that works in every case.

Finally, there is also the question of how to perform union and difference for cases in which the count is zero. For now, we ignore the zero counts during these operations, but are currently exploring strategies for incorporating them to further reduce bias and variance. Our approach is summarized in Algorithm 1 for the union and difference operations as well as the Jaccard estimate computation (the function `score`).

Algorithm 1 Homomorphic MinHash

function UNION(S_1, S_2)
  $h_{\pi_i}(S_1 \cup S_2) = \min(h_{\pi_i}(S_1), h_{\pi_i}(S_2))$
  if $c_i^{S_1} = c_i^{S_2}$ **then**
    $c_i^{S_1 \cup S_2} = c_i^{S_1} + c_i^{S_2}$
  else
    $c_i^{S_1 \cup S_2} = c_i^{S^\star}$ where $S^\star = \text{argmin}_{S_j \in \{S_1, S_2\}}(h_{\pi_i}(S_j))$

function DIFFERENCE(S_1, S_2)
  if $h_{\pi_i}(S_1) \neq h_{\pi_i}(S_2)$ **then**
    $h_{\pi_i}(S_1 \setminus S_2) = h(S_1), c_i^{S_1 \setminus S_2} = c_i^{S_1}$
  else if $h_{\pi_i}(S_1) = h_{\pi_i}(S_2)$ **then**
    if $c_i^{S_1} > c_i^{S_2}$ **then**
      $h_{\pi_i}(S_1 \setminus S_2) = h(S_1), c_i^{S_1 \setminus S_2} = c_i^{S_1} - c_i^{S_2}$
    else
      $h_{\pi_i}(S_1 \setminus S_2) > h(S_1), c_i^{S_1 \setminus S_2} = 0$

function SCORE(S_1, S_2)
  $\hat{J} = \frac{1}{\hat{n}} \sum_i \mathbb{1}\left[h_{\pi_i}(S_1) = h_{\pi_i}(S_2)\right], \forall i \quad \text{s.t.} \quad c_i^{S_1} > 0 \quad \text{and} \quad c_i^{S_2} > 0$
  $\hat{n}$ is the number of hash function pairs with non-zero counts.

E.2 Homomorphic MinHash Results

Recall that our homomorphic minhash algorithm associates a count with each minimum hash value that indicates the number of times that hash value was a unioned into the set as a minimum. During the difference operation that can occur due to a split proposal in hierarchical coreference, it is possible that these counts can become zero. Then, information is lost because it is not possible to recompute the new minimum value without traversing the entire tree, an operation that is expensive to perform in the inner loop of MCMC inference. Therefore, as a possible remedy, we propose to ignore hash values that have a count of zero in certain situations. In certain situations, this is unbiased[5] A primary concern though,

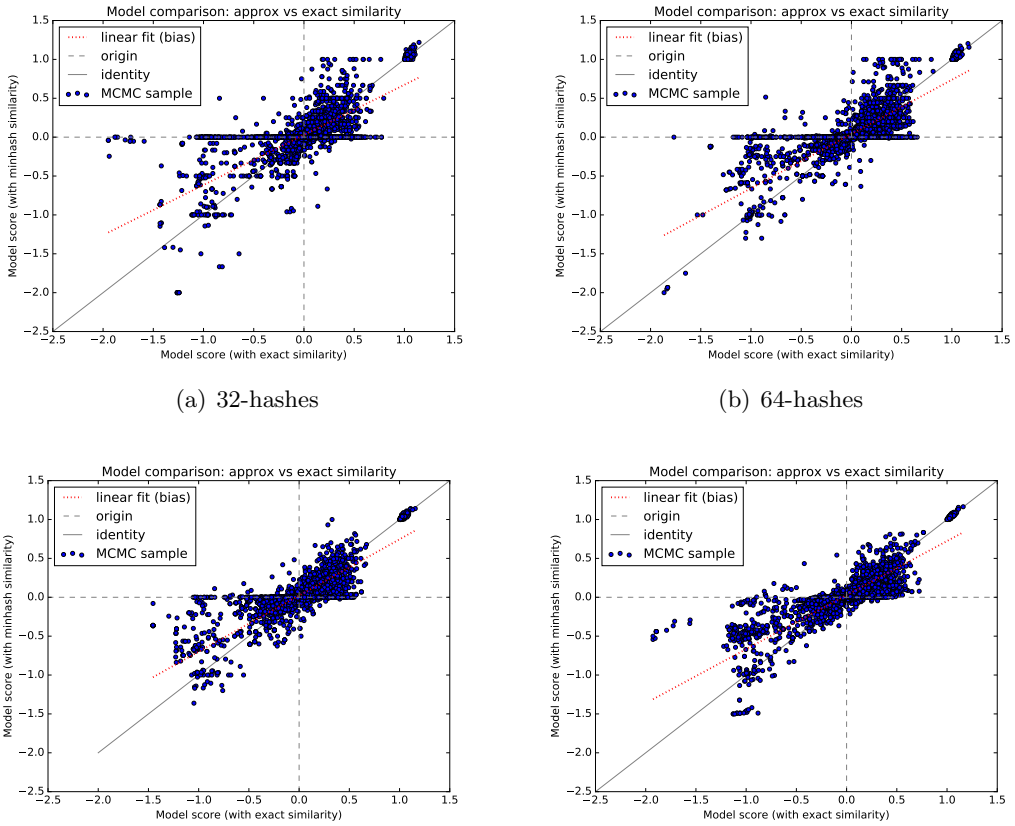
5. in other cases it is biased and we should recompute the values in these cases, but we save this for future work.

is that as the number of samples increases, the number of entries with a zero count will increase and (a) the variance of the estimate might increase and (b) the estimate might be impossible to compute if all counts become zero.

We indirectly study (a) by comparing our homomorphic minhash estimates to the exact Jaccard estimate in much the same we did for simhash (Figure 7). Each point in the plot represents an MCMC sample, which we gather by running the exact model with no compressive data structures as before. The x-axis is the difference in the exact Jaccard similarity estimates of the context bags and the y-axis is the difference in exact Jaccard similarity between the homomorphic minhash representations of these bags. Unlike simhash, minhash employs multiple hash functions; therefore, we fix the number of bits to 64 and vary the number of hash functions instead. We also fit a linear model to the points in order to help assess the bias (as seen by how far off it is from the reference line), and find that indeed our method introduces some bias. We further examine the percentage of hash functions have a non-zero count. In particular, after running hierarchical coref in the previous experiment for 100,000 samples, we look at the root level entities and measure what percentage of their hash functions have counts that have gone to zero). The reason we look at root entities only is that they are the most likely to have had difference operations applied to them since every time MCMC proposes to split a sub-tree, the root must undergo a difference operation. We find that on average about 16% of these hash-values have a non-zero count, and find that 78% or these representations have fewer than 25% non-zeros, 15% have between 25-50% non-zeros, 4% have between 50-75% and the rest have above 75%.

As for (b), we check after each sample if it results in a hash representation for which all its counts are zero, thus making Jaccard estimates impossible.[6] We record for each of the 100,000 samples whether or not the sample results in a hash for which all counts are zero. We find that this only happens for the 32 and 64 hash function variants, and even in these cases, the percentage is small (1.4 and 1.7% respectively), and they are responsible for the regimentation of points along the x-axis in Figures 7(a)–7(b). For 128 and 256 hash function variants, this situation never occurs. This is encouraging as it means, from the perspective of being able to compute an estimate, that we rarely need to refresh the values.

6. Actually, this is not entirely true since some estimates are possible because not all information is lost. Technically a count of zero indicates that although we do not know what the minimum is, we know what it is *not*, viz., the hash value associated with the count. Therefore, if it happens to be the case that the representation we compare against (a) contains non-zeros counts everywhere and (b) has a hash an identical hash-value for each hash function, we can conclude the Jaccard is zero. Our implementation supports this case as a special case, but it is so unlikely to occur.



(a) 32-hashes

(b) 64-hashes

(c) 128-hashes

(d) 256-hashes

Figure 7: Model score comparison with homomorphic minhash and exact sparse-vector representations. The closer points are to the identity reference line, the better.